FREE Win a ZX Printer and Software!

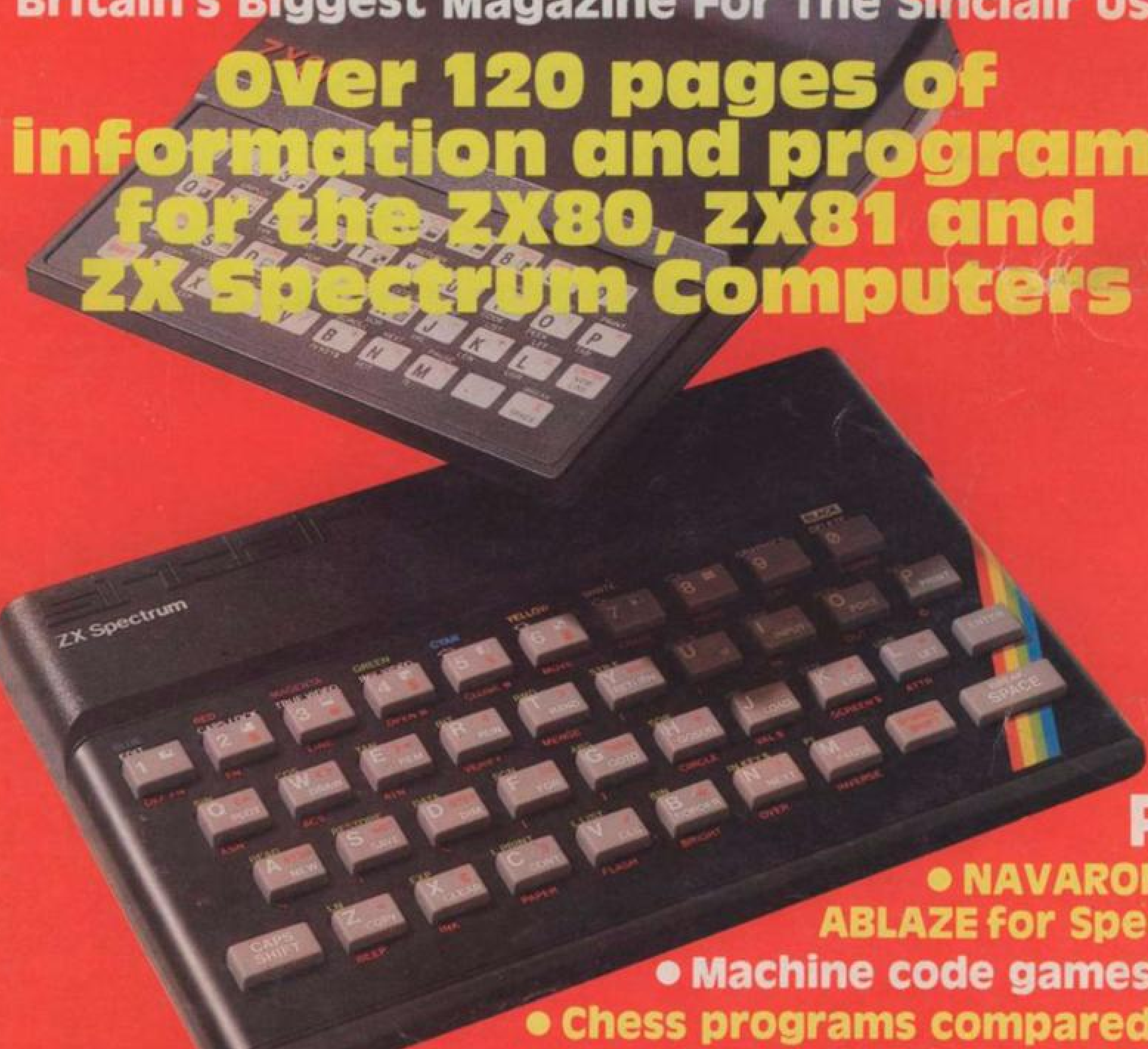
ZX COMPUTING

OCT/NOV 1982
£1·75

MACHINE CODE
PT 1
93

Britain's Biggest Magazine For The Sinclair User

Over 120 pages of information and programs for the ZX80, ZX81 and ZX Spectrum Computers



PLUS

- NAVARONE ABLAZE for Spectrum
- Machine code games to try
- Chess programs compared
- Business and education with your Sinclair computer
- Add a keyboard to your Spectrum, ZX80 or ZX81

ZX99 AUTOMATIC TAPE CONTROLLER FOR THE SINCLAIR ZX81

● DATA PROCESSING

The ZX99 gives you software control of up to four tape drives (two for reading, two for writing) allowing merging of data files. This is achieved by using the remote sockets of the tape drives, controlled by USR statements or commands.

● RS232C INTERFACE

The ZX99 has an RS232C output allowing connection with any such printer using the full ASCII character code (you can now print on plain paper in upper or lower case, and up to 132 characters per line) at a variable baud rate up to 9,600

● SPECIAL FEATURES

There are so many special features it is difficult to list them all, for example:

AUTOMATIC TAPE COPY: You can copy a data file regardless of your memory capacity as it is processed through the Sinclair block by block.
TAPE BLOCK SKIP: Without destroying the contents of RAM
DIAGNOSTIC INFORMATION: To assist in achieving the best recording settings.

The ZX99 contains a 2K ROM which acts as an extension to the firmware in the Sinclair ROM. The ZX99's ROM contains the tape drive operating system and the conversion to ASCII for the RS232C output.

There is an extension board on the rear to plug in your RAM pack (larger than 16K if required). The unit is supplied with one special tape drive lead, more are available at £1 each.

NOW ONLY
£49.95
PLUS
£2.95 p&p



● ZX99 SOFTWARE

We now have available "Editor 99", a quality word processing program including mail-merge, supplied on cassette for £9.95. Also following soon:

- **Stock Control (October)**
- **Sales Ledger (November)**
- **Business Accounts**
- **Debtors Ledger**
- **Tax Accounting**

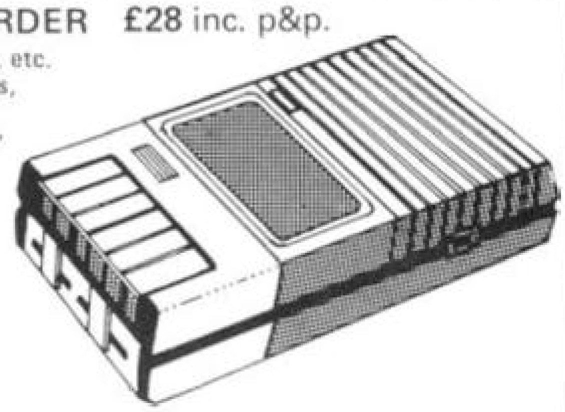
Dept. ZX2 Data – Assette, 44 Shroton Street, London NW1 6UG. 01-258 0409

---

data-assette®
44, Shroton Street
London NW1
Tel 01-258 0409

● FERGUSON CASSETTE RECORDER £28 inc. p&p.

Tested with ZX81, Acorn, BBC, Dragon etc. etc.
Features: Din, Ear, Mic. and Remote sockets, Tape Counter, Tone Control, Built-in Mic., Autostop. Battery/mains. Recommended by Acorn for use with BBC computer.



● STAR DP8480

RS232C (SERIAL) **£285** plus £6 Securicor delivery
CENTRONICS (PARALLEL) **£265** plus £6 Securicor delivery

This professional printer works with almost any computer with very good upper and lower case typeface.

- Bi-directional
- 80 column width (10" paper)
- Switchable – Tractor or Friction Feed
- 80 chrs. per second



● 2,000 SHEETS OF PRINTER PAPER

£19.50 plus £3.50 p&p.

● E690 REVOLVING CASSETTE RACK



Single – **£2.99** (holds 32 tapes or 20 in cases)
Double – **£5.99** (holds 64 tapes or 40 in cases)
Treble – **£8.99** (holds 96 tapes or 60 in cases)
Quad – **£11.99** (holds 128 tapes or 80 in cases)
All plus £1 p&p.

● BBC CASSETTE LEAD

7 pin plug to two 3.5mm plugs and one 2.5mm plug. Only **£2** inc. P & P.
Other leads available – please telephone.

COMPUTER CASSETTES

High quality, screw assembled cassettes supplied with library boxes. Any lengths available.

C5 – 35p C10 – 37p C12 – 38p
C15 – 39p C20 – 41p C25 – 43p
C30 – 44p

P&P 10% (min. charge £1.50)

ORDER FORM

Dept. ZX2 Data – Assette, 44 Shroton Street, London NW1 6UG. 01-258 0409

Code	Item	No.	Price	P & P	Total

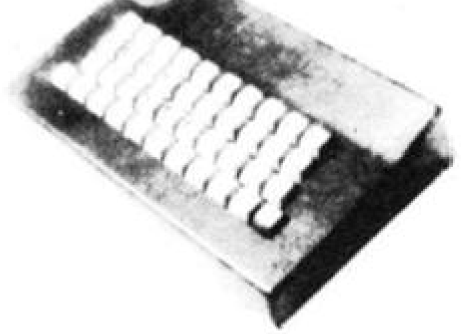
Cheques/PO made payable to Storkrose Ltd.
Charge my Access/Visa No. ....
Signed ____________ Name ____________
Address ____________

KAYDE Electronic Systems Ltd

ZX KEYBOARDS FULLY CASED

ZX81 WITH REPEAT KEY

This is a highly professional keyboard using executive buttons as found on top quality computers. It has a repeat key and comes complete in its own luxury case. This is a genuine professional keyboard and should not be confused with toy keyboards currently available on the market.
As reviewed by Tim Hartnel ZX Computing/Interface **£37.95**



NEW SPECTRUM

This is a highly professional keyboard, taken from our World Famous and well proven 81 keyboard, it has been redesigned to complement your ZX Spectrum.
BUY NOW BEFORE THE RUSH **£37.95**

FULLY CASED ZX81

RAMPACKS

MASSIVE ADD ON MEMORY

NEW Spectrum

MEMORIES YOU CAN RELY ON

64K **£72.95**

16K **£29.95**

Up grade your Spectrum now to 48K. Just add our 32K memory extension plug in board to your Spectrum which already has 16K to give you a massive 48K of memory.
£48.95

ZX81 4K GRAPHICS BOARD

The *KAYDE* Graphics Board is probably our best accessory yet. It fits neatly inside your ZX81. It comes complete with a pre-programmed 4K Graphics ROM. This will give nearly 450 extra graphics and with the inverse makes a total of nearly nine hundred.

The *KAYDE* Graphics Board has facilities for either 2K of RAM (for user definable graphics) 4K of ROM or our 4K Tool Kit Chips that will be available shortly. All the graphics are completely software controlled therefore they can be written into your programmes. Here are a few examples:
A full set of space invaders — Puckman — Bullets, Bombs — Tanks — Laser Bases and Alien Ships
NO EXTRA POWER NEEDED **£29.95**

All Hardware comes fully built and tested and complete with a 14 day money back option

FLEXIBLE RIBBON CONNECTOR

Stops movement of RAM PACK £12.95
IN/OUT PORT £10.95
MOTHER BOARD
Complete with 5 volt regulator **£18.95**

16K GRAPHICS BOARD SOFTWARE

PECKMAN The only true ZX version of the popular arcade game.
SPACE INVADERS The best version available anywhere.
CENTIPEDE "In all I think this is the best presented moving graphic programme I've yet seen" — Phil Garratt, INTERFACE **£5.95 EACH**
Graphic Software can only be used with a Graphics board

16K 81 SOFTWARE

CENTIPEDE "In all, I think this is the best presented moving graphic programme I've yet seen" Phil Garratt — Interface.
3D 3D LABYRINTH A cubit maze that has corridors which may go left, right, up and down. **£5.95**

4K Tool Kit full of utilities to aid the programmer in constructing and de-bugging E. PROM version for use with graphics ROM. **£9.95**

VIC-20 SOFTWARE NEW

THE KAYDE VALLEY The ultimate in adventure games

Othello Plus many more. **£7.95**

VIC 20 dealer

Qty	Item	Item price £	Total £
	ZX Spectrum Cased Keyboard s	37.95	
	ZX Spectrum 32K Rampack	48.95	
	ZX81 Cased Keyboard with repeat key	37.95	
	ZX81 64K Rampack	72.95	
	ZX81 16K Rampack	29.95	
	ZX81 4K Graphics Board	29.95	
	ZX81 Flexible Ribbon Connector	12.95	
	ZX81 Mother Board	18.95	
	ZX81 In out port	18.95	
	ZX81 16K Graphic Software	5.95	
	ZX81 16K Software	5.95	
	Vic 20 Software	7.95	
	Name of Software	- p&p	

*I enclose a cheque postal order payable to *KAYDE Electronic Systems Ltd.*, for £
*Please charge to my Access Barclaycard Trustcard account no.
*Please delete complete as applicable.

Please print

Signature

Name: Mr Mrs Miss

Address

Please allow £1.50 P&P for all Hardware 50p for all software (Dept ZX)

SEND FOR A FREE CATALOGUE STATING TYPE OF COMPUTER.

WHY WAIT TO PAY MORE
FAST IMMEDIATE DELIVERY

Post To **KAYDE ELECTRONIC SYSTEMS LTD**
Dept **THE CONGE, GREAT YARMOUTH**
NORFOLK NR30 1PJ
Tel: 0493 57867 (Dept ZX)
Telex 957247 CHATCOM G

Don't forget you can always order on the telephone with your credit card *Dealers welcome*

ZX COM

ZX Computing
Vol. One
Number Three
Oct/Nov 1982

Editor: Tim Hartnell
Editorial Assistant: Helen Bruff
Advertising Manager: Neil Johnstone.
Managing Editor: Ron Harris
Managing Director: T J Connell

Origination and design by MM Design & Print, 145 Charing Cross Road, London WC2H OEE.

Published by Argus Specialist Publications Ltd, 145 Charing Cross Road, London WC2H OEE.

CONTENTS

ZX Computing is constantly on the look-out for well written articles and programs. If you think that your efforts meet our standards please feel free to submit your work to us for consideration.
Material should be typed if possible. Any programs submitted must be listed, cassette tapes alone will not be accepted, and should be accompanied by documentation to explain how they work and make it easy to run them. All submissions will be acknowledged. Any published work will be paid for.
All work for consideration should be sent to the Editor at our Charing Cross Road address.

MPUTING

ZX Computing is published bi-monthly on the fourth Friday of the month. Distributed by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS. 01-247 8233. Printed by: Henry Garnett Ltd., Rotherham.

tum



The ZX Printer – available now

Designed exclusively for use with the Sinclair ZX range of computers, the printer offers ZX Spectrum owners the full ASCII character set – *including lower-case* characters and high-resolution graphics.

A special feature is COPY which prints out exactly what is on the whole TV screen without the need for further instructions. Printing speed is 50 characters per second, with 32 characters per line and 9 lines per vertical inch.

The ZX Printer connects to the rear of your ZX Spectrum. A roll of paper (65ft long and 4in wide) is supplied, along with full instructions. Further supplies of paper are available in packs of five rolls.



The ZX Microdrive – coming soon

The new Microdrives, designed especially for the ZX Spectrum, are set to change the face of personal computing.

Each Microdrive is capable of holding up to 100K bytes using a single interchangeable microfloppy.

The transfer rate is 16K bytes per second, with average access time of 3.5 seconds. And you'll be able to connect up to 8 ZX Microdrives to your ZX Spectrum.

All the BASIC commands required for the Microdrives are included on the Spectrum.

A remarkable breakthrough at a remarkable price. The Microdrives are available later this year, for around £50.



RS232/network interface board

This interface, available later this year, will enable you to connect your ZX Spectrum to a whole host of printers, terminals and other computers.

The potential is enormous. And the astonishingly low price of only £20 is possible only because the operating systems are already designed into the ROM.

ZX Spectrum

Available only by mail order and only from

sinclair

Sinclair Research Ltd,
Stanhope Road, Camberley,
Surrey, GU15 3PS.
Tel: Camberley (0276) 685311.

How to order your ZX Spectrum

BY PHONE – Access, Barclaycard or Trustcard holders can call 01-200 0200 for personal attention 24 hours a day, every day. BY FREEPOST – use the no-stamp needed coupon below. You can pay by cheque, postal order, Barclaycard, Access or Trustcard.

EITHER WAY – please allow up to 28 days for delivery. And there's a 14-day money-back option, of course. We want you to be satisfied beyond doubt – and we have no doubt that you will be.

To: Sinclair Research, FREEPOST, Camberley, Surrey, GU15 3BR. **Order**

Qty	Item	Code	Item Price £	Total £
	Sinclair ZX Spectrum – 16K RAM version	100	125.00	
	Sinclair ZX Spectrum – 48K RAM version	101	175.00	
	Sinclair ZX Printer	27	59.95	
	Printer paper (pack of 5 rolls)	16	11.95	
	Postage and packing: orders under £100	28	2.95	
	orders over £100	29	4.95	
			Total £	

Please tick if you require a VAT receipt ☐

*I enclose a cheque/postal order payable to Sinclair Research Ltd for £ ______

*Please charge to my Access/Barclaycard/Trustcard account no. ______

*Please delete/complete as applicable

Signature ______

PLEASE PRINT

Name: Mr/Mrs/Miss ______

Address ______

ZXC 810

FREEPOST – no stamp needed. Prices apply to UK only. Export prices on application.

Welcome

Welcome to the third issue of *ZX Computing*. The Spectrum has certainly set the ZX world on fire, and in this issue we bring you more articles and programs designed to help you develop your programming skills on the Spectrum.

We've not neglected the ZX81 nor the ZX80, and you'll find these computers are well represented in this issue.

Before I outline the contents of the issue for you, I'd like to draw your attention to our competition. We're giving away a ZX printer, and five sets of the ZX81 software which Uncle Clive is selling. The winner of the competition gets a printer, plus a set of software, and there are four prizes of software for the runners-up. Details of the competition are included in the 'News' section of this issue.

When you compare the published software in magazines today compared with the programs which were published back at the beginning of ZX history, you see how much programming standards have improved. In this issue, we have an article on 'structured programming' which may well help you to improve your own standards of program writing.

Tim Roger's program PROPORTIONAL SPACING is designed to solve the problem of messy word output. The program not only ensures that words are not split at the end of lines, but also 'pads out' each line in order to use all 32 characters across.

This program demands a 16K ZX81, as does the one by N G Strong, of Epsom, an engineer who retired before the advent of personal computers and even pocket calculators.

The ability of the ZX81 to SAVE and LOAD a named program on tape opens up many interesting possibilities. James Calderwood, from Coleraine, explains how it is possible to load a program from a directory, by just entering the number printed beside the program of your choice in a menu on the screen.

So that things don't get too serious, this issue of *ZX Computing* has a number of very fine games for you to play, including a well-written GRAND PRIX program from Jim Archer, of Frimley, Surrey.

Another great game is FOX AND CHICKEN, written by Jim McCartney of Coleraine. This program is a mixture of machine code and BASIC, and examining the listing should help you further develop your programming skills. Stephen Adams looks at the electronics beneath ZX keyboards, and tells you how to add an extension keyboard of your own.

If you're bogged down with a bug in your program, a flowchart can help. Henry Budgett, editor of 'Computing Today', tells you how to go about flowcharting in your relentless search for bugs.

In our last issue of *ZX Computing*, we introduced a new section — ZX Education — and this has proved very popular, so we bring you a slightly expanded edcuational section in this issue. If your school uses ZX computers in any way, why not write into the magazine and let us know. We'd love to be able to run a photograph of you and your classmates using their ZX computers, and share your ideas for using the computers in education.

We make sure that every issue of the magazine has several major articles and programs, as well as a number of shorter ones, so that those who want a really meaty program will have something to get their teeth into. One of the programs which fits this classification in this issue is a superb 7K program for the ZX81 — SLOT MACHINE — written by Adam Waring and Mike Cleverley of Hull. The program uses a flashy machine code routine to reverse the display when you win . . . and at the end of the game when you lose.

Another 'heavy' article comes from Thomas Ballantyne, Paisley, Scotland, who has written a program devised to calculate and illustrate, using circuit and phasor diagrams, the characteristics of a series A.C. electrical circuit.

Contributions

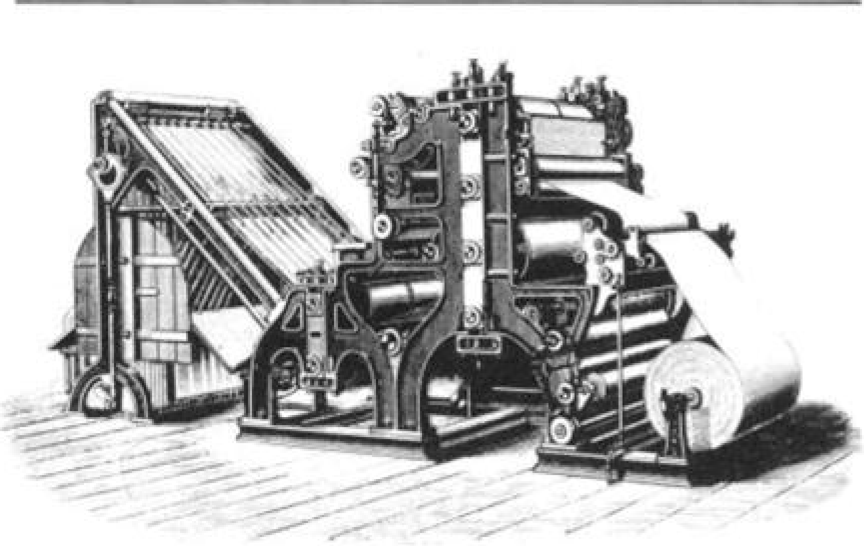
We're on the lookout now for good programs and articles for the next issue of *ZX Computing*. Program listings are vital (rather than just providing the program on cassette), along with clear instructions on what the program does, how it works, and what the user will see when he or she runs it.

Any kinds of programs are useful, but we are particularly interested in ones which use ZX BASIC in particularly clever ways, or in ones which contain routines which can be re-used in other programs.

It is vital that the programs you send us are totally original, and not 'adapted' from programs in other magazines, or in books. I've even had two programs submitted to me here at *ZX Computing* which were 'borrowed' from my own books.

All contributions we use are paid for, of course, so if you'd like to contribute to forthcoming issues and make a bit of pocket money, look through the contents of this issue, and if you can write as well, or better than, our present contributors, let's hear from you. But for now, get down to enjoying this issue with your ZX computer.

TIM HARTNELL, Editor





Dear ZX Computing,
I am writing to you about your DODGEM CHOMP program in issue two which is very entertaining and could give hours of fun. However, in the introduction, you state that the game could go on for ever, and that Tim Rogers' high score was 450. If, however, he had managed to attain about 500, he would have seen that, as the listing stands, the maximum possible is roughly 580 or so, because of erroneous logic in lines 900 to 940. The following corrections allow the game to carry on indefinitely:

```
920 LET P = G
930 LET G = H
940 LET H = P
```

Andrew Goodsell,
Penge



Machine code

Dear ZX Computing,
I have just been reading your second issue of ZX Computing and find it a very useful and interesting source of data.

I did, however, note an error which I felt I should bring to your attention as it would cause difficulty to your readers.

In the "breaking out" program there is a coding 4A 8440 used in the 3rd line of the machine code routine. This doesn't make sense as 4A is LD C, D and the 8440 has no meaning there. The correct code would be 3A 8440 which is LD A, (4084h 16516d).

Ralph Hilton,
Folkestone, Kent

Memory query

Dear ZX Computing,
I have read reports that the 16K Spectrum uses 7K to provide the colour and graphics, leaving only 9K of "usable" memory. There are some marvellous Adventure programs around for the ZX81 which need the 16K RAMpack. As a layman about to buy his first computer, am I right in thinking that these — when translated into Spectrum BASIC, will not fit into what is left of its standard memory and that I would therefore need the 48K version to take them? I appreciate the 48K would give me much more scope as more involved software is developed but it is £50 more initially.

The ZX81's 16K RAMpack has dropped £20 as chips have become cheaper. Is it likely that if I bought a 16K Spectrum, the £60 cost to up-grade it to 48K might be reduced in the future?

I have already seen one ad. for a 64K RAMpack to stick-on the back to give it 80K. If I bought such a non-Sinclair add-on, do I just plug it in and carry on? I have read articles (which I profess not to wholly understand) which say the machine does not know how much memory it has got (if non-standard) and starts throwing in phrases like PEEK, POKE and altering RAMTOP.

Please answer in basic ENGLISH and not English BASIC!

Richard Carsons,
Ewell, Surrey

● *Horrors! Do not, repeat not, connect anything except the ZX printer to the back of the Spectrum (apart from products specifically produced for the Spectrum). You will damage both the computer and the add-on memory if you plug memory designed for the ZX81 into the Spectrum. Many programs which are marked '16K' really should be marked 'more than 1K', as few so-called 16K programs actually use all the available memory. However, an ADVENTURE program is likely to use just about all the available memory, so a ZX81 ADVENTURE program designed for a 16K ZX81 is not likely to fit on a Spectrum. There has been no discussion, apart from the information in the leaflet, about the 'upgrade' from a 16K to a 48K but it is possible that private firms may in due course be offering this service at a lower price than the Sinclair one.*



No speaka da latin

Dear ZX Computing,
I enjoyed your 'Pig-Latin Translator' in the last issue of ZX Computing, and thought you might be interested in seeing the results of my taking up the challenge to write a 'Pig-Latin Translator'. All the translating is done in lines 40 and 45.

The second program is my own 'Latin translator', which produces different results to the one you published in your last issue. Lines 12, 14 and 16 are the endings of words. Line 40 forms the Latin text, and the subroutine puts UM or US onto the end of a word if it ends in ING, ER or ND.

E P Whitby,
Chilwell, Notts

● *Thanks very much for the programs. They are certainly a good development from the one we published in issue two. We're always interested in seeing developments of the programs published in ZX Computing.*



```
   5 REM A PIG LATIN TRANSLATOR
   6 REM (C) E.P.WHITBY
              JULY 25 1982
  10 PRINT "ENTER MESSAGE","WORD
BY WORD"
  20 PRINT "ENTER $ TO  STOP "
  25 PRINT
  30 INPUT A$
  35 IF A$="$" THEN STOP
  40 LET B$=A$( TO LEN A$-2)
  45 LET T$=A$(LEN A$-1)+B$
  50 PRINT T$;" "
  55 IF RND>.7 THEN PRINT
  60 GOTO 30
```

```
   5 REM DOG LATIN
   6 REM (C) E.P.WHITBY
              JULY 25 1982
  10 DIM A$(3,2)
  12 LET A$(1)="O"
  14 LET A$(2)="UM"
  16 LET A$(3)="US"
  20 LET T=INT (RND*3)+1
  25 INPUT T$
  30 IF LEN T$>3 THEN GOSUB 100
  35 IF T$="$" THEN STOP
  40 LET T$=T$+A$(T)
  45 PRINT T$;" ";
  50 IF RND>.7 THEN PRINT
  55 GOTO 20
 100 LET E$=T$(LEN T$-2)
 110 IF E$="ING" THEN LET T=3
 120 IF E$="ER" OR E$="ND" THEN
LET T=2
 130 RETURN
```



Bouquets

Dear ZX Computing,
Just a note to congratulate you on the contents of your second issue. Even the printing had improved, apart from the VAT programs in the business section.

Keep up the good work . . . and I'll keep on buying it.

Martin Shaftesbury,
Wilmslow, Cheshire



Improving your tiling

Dear ZX Computing,
The program 'tile crazy' by K Mahogany in issue 2 of ZX Computing had a number of faults to my way of thinking.

First, it asked for the destination of each move when, in fact, there is always only one valid destination — the square from which the previous move was made.

Second, it did not check that each move was being made from a square adjacent to the empty one. It was easy to cheat!

Third, it always started from the same position as set up in a pair of DATA statements. I am referring here to the Spectrum version, of course.

Finally, that initial position was such that it was impossible to achieve the correct alphabetical order by legal moves! The best that could be achieved was an order in which one pair of letters was interchanged and, as most readers will probably be aware, in this type of puzzle, it is only possible to interchange an even number of pairs of letters.

My version of the program (for the Spectrum) corrects these faults and a printout is enclosed, and below I detail the changes I have made.

In the data statement on line 430, one pair of letters has been interchanged to give a valid starting position. Also the space, represented by -32, has been moved to the end of the statement to give a known starting position for the program to work from. This does not affect the validity of the order of the letters.

The RETURN at line 410 has been changed to GOTO 500. At line 500 onward coding has been added to shuffle the initial position by performing a random, even number of interchanges of pairs of letters randomly selected leaving square 16 empty. This shuffle preserves the validity of the initial state.

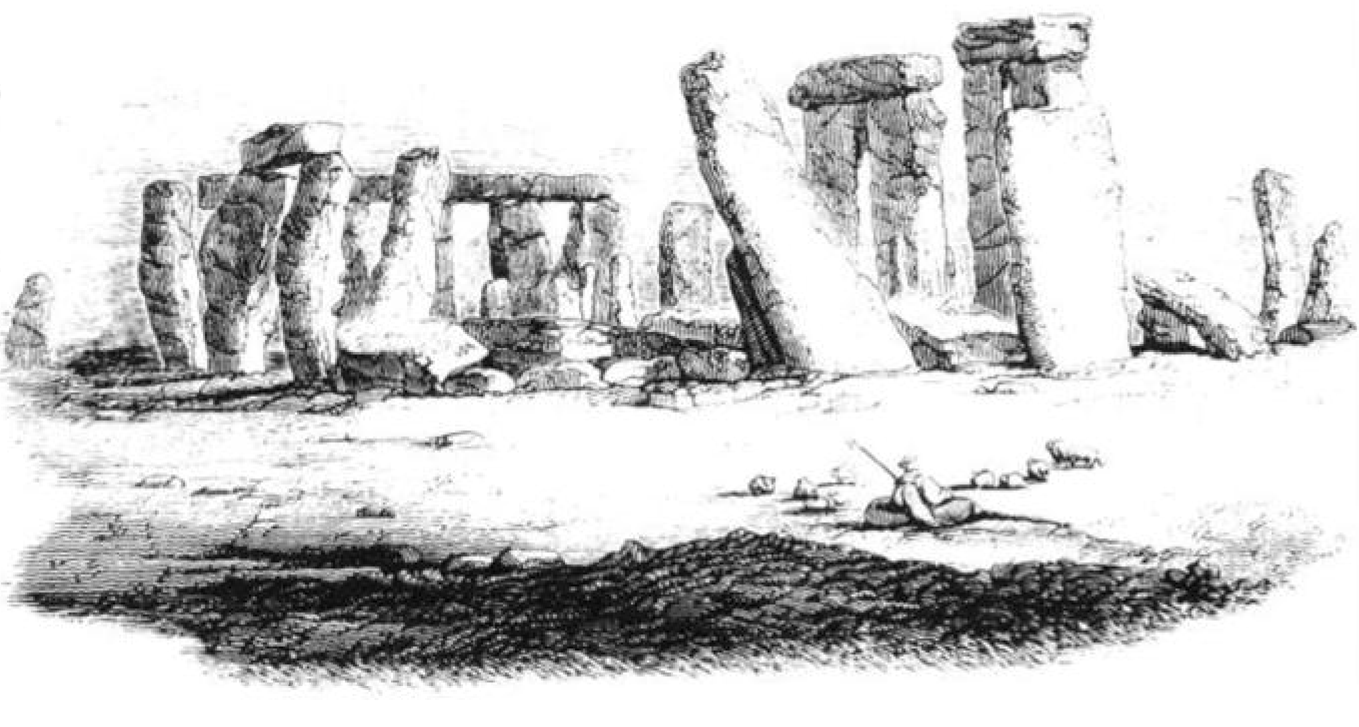
Variable e is set to 16 in line 360 and is subsequently used for the destination of the requested move in lines 100 and 110 and updated to the new empty square in line 115.

In line 70 the program tests that the requested move origin is orthogonally adjacent to the empty square.

Some cosmetic changes have also been made to the PRINT statements in lines 230-260 but these were just to satisfy personal preferences.

P G Moulton,
Leigh-on-Sea, Essex

```
  10 REM tiling
  30 GO SUB 330
  40 GO SUB 200
  50 GO SUB 200
  60 INPUT INK 7;"which one to m
ove7 ";x
  70 IF ABS (x-e)<>4 AND ABS (x-
e)<>1 THEN GO TO 60
 100 LET a(e)=a(x)
 110 LET a(x)=32
 115 LET e=x
 120 LET go=go+1: GO TO 50
 200 PRINT AT 0,3; PAPER 7; INK
3;"go number "; INK 2;go: PRINT
: PRINT
 230 PRINT INK 8;"   ";CHR$ a(1)
;CHR$ a(2);CHR$ a(3);CHR$ a(4),"
  1  2  3  4"
 240 PRINT INK 8;"   ";CHR$ a(5)
;CHR$ a(6);CHR$ a(7);CHR$ a(8),"
  5  6  7  8"
 250 PRINT INK 8;"   ";CHR$ a(9)
;CHR$ a(10);CHR$ a(11);CHR$ a(12
),"    9 10 11 12"
 260 PRINT INK 8;"   ";CHR$ a(13
);CHR$ a(14);CHR$ a(15);CHR$ a(1
6),"   13 14 15 16"
 270 RETURN
 330 REM  initialize
 340 DIM a(16)
 350 FOR b=1 TO 16: READ m: LET
a(b)=m+64: NEXT b
 360 LET e=16
 390 LET go=1
 400 PAPER 6: BORDER 2: CLS
 410 GO TO 500
 420 DATA 9,14,5,2,11,6,1,4,12
 430 DATA 7,10,13,8,15,3,-32
 500 LET n=(RND*20+1)*2
 510 FOR i=2 TO n
 520 LET x=RND*15+1
 530 IF a(x)=32 THEN GO TO 520
 540 LET y=RND*15+1
 550 IF x=y OR a(y)=32 THEN GO T
O 540
 560 LET j=a(x)
 570 LET a(x)=a(y)
 580 LET a(y)=j
 590 NEXT i
 600 RETURN
```



MEMOTECH Explores the Excellence of your ZX81



Memotech's Memopak Range

All five of the currently available Memopaks are housed in elegant black anodised aluminium cases, and are styled to fit wobble-free onto the back of the ZX8I, allowing more add-ons (from Memotech or Sinclair) to be connected.

£68.70 plus VAT

MEMOPAK 64K MEMORY EXTENSION

The 64K Memopak extends the memory of the ZX81 by 56K, and with the ZX81 gives 64K, which is neither switched nor paged and is directly addressable. The unit is user transparent and accepts commands such as 10 DIM A(9000).

Breakdown of memory areas...0-8K-Sinclair ROM. 8-16K-This area can be used to hold machine code for communication between programmes or peripherals. 16-64K-A straight 48K for normal Basic use.

£43.43 32K / £26.00 16K plus VAT

MEMOPAK 32K and 16K MEMORY EXTENSIONS

These two packs extend and complete the Memotech RAM range (for the time being!) A notable feature of the 32K pack is that it will run in tandem with the Sinclair 16K memory extension to give 48K RAM total.

£52.00 plus VAT

MEMOPAK HIGH RES GRAPHICS PACK

HRG Main Features — • Fully programmable Hi-Res (192 × 248 pixels) • Video page is both memory and bit mapped and can be located anywhere in RAM. • Number of Video pages is limited only by RAM size (each takes about 6.5K RAM) • Instant inverse video on/off gives flashing characters • Video pages can be superimposed • Video page access is similar to Basic plot/unplot commands • Contains 2K EPROM monitor with full range of graphics subroutines controlled by machine code or USR function

£34.70 plus VAT

MEMOPAK CENTRONICS TYPE PARALLEL PRINTER INTERFACE

Main Features — • Interfaces ZX8I and parallel printers of the Centronics type • Enables use of a range of dot matrix and daisy wheel printers with ZX8I • Compatible with ZX8I Basic, prints from LLIST, LPRINT and COPY • Contains firmware to convert ZX8I characters to ASCII code • Gives lower-case characters from ZX8I inverse character set

Coming Soon...



A complete range of ZX81 plug-in peripherals

Digitising Tablet **RS232 Interface**

We regret we are as yet unable to accept orders or enquiries concerning the above products, but we'll let you know as soon as they become available.

Access/Barclaycard Holders Please Tel: Oxford (0865) 722102 (24 Hrs)

Please make cheques payable to MEMOTECH Ltd.

Please Debit my Access/Barclaycard* account number

*Please delete whichever does not apply

Please send me	Price	No	Total
64K RAM £68.70 + £10.30 VAT	£79.00		
32K RAM £43.43 + £6.52 VAT	£49.95		
16K RAM £26.00 + £3.90 VAT	£29.90		
HRG £52.00 + £7.80 VAT	£59.80		
CENTRONICS I/F £34.70 + £5.20 VAT	£39.90		
Packaging & Postage £2.00 per unit			
		TOTAL ENC	

SIGNATURE ____________ DATE ____________

NAME ____________ ADDRESS ____________

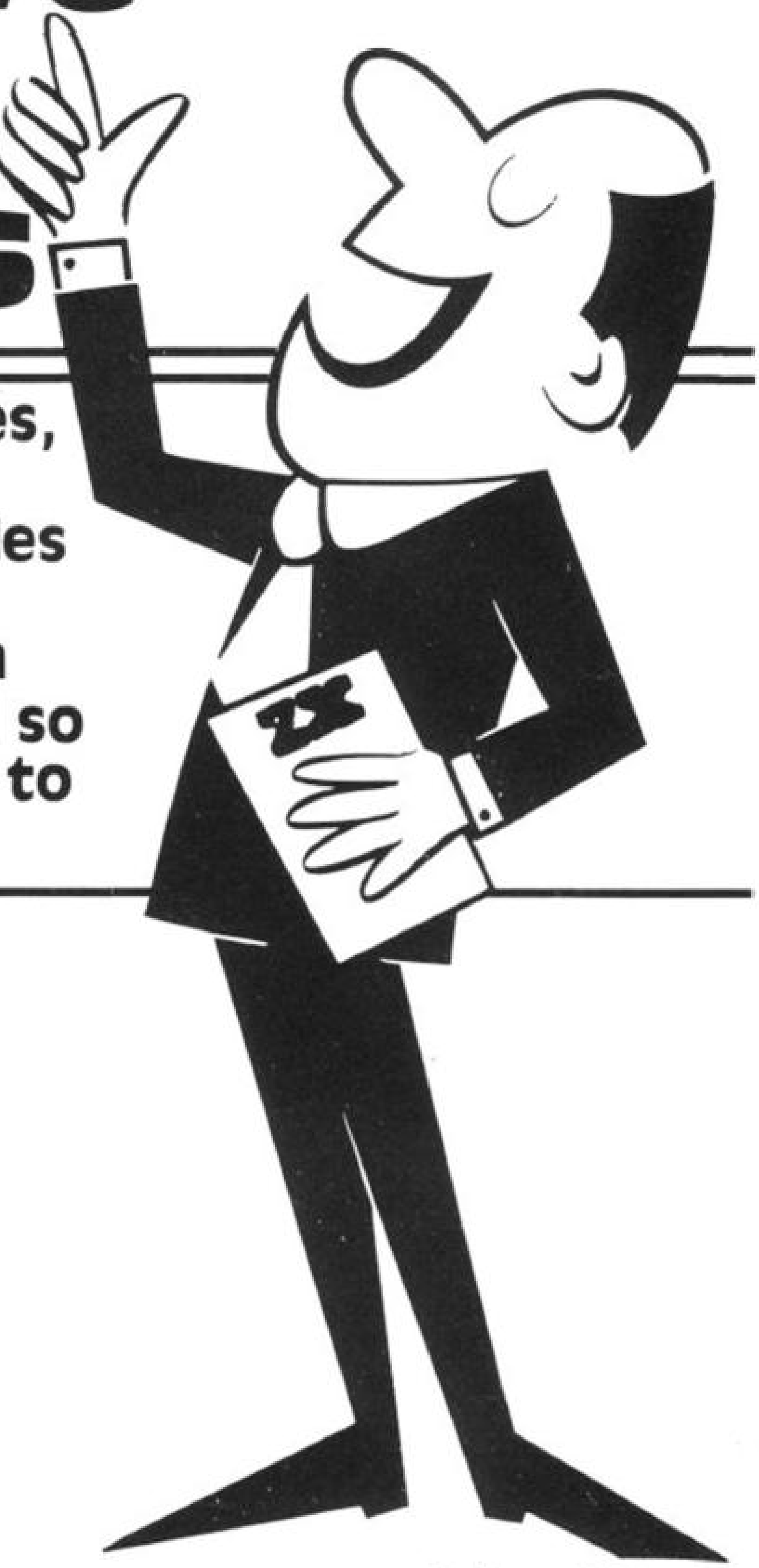
____________ TELEPHONE ____________

© R. Branton & G.A.C. Boyd 1982

We want to be sure you are satisfied with your Memopak – so we offer a 14-day money back Guarantee on all our products.

Memotech Limited, 3 Collins Street, Oxford OX4 1XL, England Tel: Oxford (0865) 722102 Telex: 837220 Orchid G

More hints 'n' tips to improve your programs



In our last issue, Dilwyn Jones, an experienced ZX programmer from North Wales shared a number of useful techniques for working with the ZX81. This article proved so popular, we've asked Dilwyn to pass on a few more ideas.

Suppose you wanted a character array to hold the names of the months. There are twelve months in one year and the name of the longest month is SEPTEMBER, which consists of nine letters. On your computer you would say:

```
10 DIM A$(12,9)
```

to give you an array of twelve words each up to nine letters long. READ/DATA would be very useful to assign the names to the array, but the computer does not have this facility. So you would probably end up doing this:

```
20 FOR N=1 TO 12
30 INPUT A$(N)
40 NEXT N
```

and the variables could be saved on tape along with the program once you've entered all the data into the array. When you came to use the array you would find that names which were less than nine letters long had been stretched out with spaces at the end to make them nine letters long to fit the array. So if you had the line

```
500 PRINT A$(5); "IS THE MONTH OF YOUR BIRTHDAY"
```

you would end up with MAY   IS THE MONTH OF YOUR BIRTHDAY All those extra spaces are ugly – it might not bother you with a word like DECEMBER, where you would get only one extra space, but with the word MAY you get six extra unwanted spaces, so we need to ensure that any trailing spaces (spaces after the word) are not PRINTed. Here is a routine to do this.

You will need to specify which part of the array is used – which word if you like – and this is represented by an X in the listing. Add these lines to the ones above:

```
490  INPUT X
500  GOSUB 6000
510  PRINT A$(X, TO A);" IS
     THE MONTH OF YOUR
     BIRTHDAY"
520  STOP
8000 FOR A=LEN A$(X) TO 1
     STEP -1
8010 IF A$(X,A)<>" " THEN
     RETURN
8020 NEXT A
8030 RETURN
```

RUN the program and enter the names of the months one by one in order. As an experiment, try leaving one month as all spaces (just press NEWLINE for one name). You might expect an error to arise if A$(X) is composed entirely of spaces, but this is all catered for. If this does happen then A will be 0 and A$(X, TO A) will be A$(X,1 TO 0) which you might expect to give a subscript error. But the computer, as we've seen, has a special interpretation for this kind of expression (where the first figure in a string slice is larger than the second), you will get the empty string, so it seems you don't have a birthday.

One small note. Look at line 510. It looks as though there's a number missing before TO. This means the same as A$(X,1 TO A) because if you leave out the number before TO the computer will assume you meant 1. Don't forget to include the comma before TO.

Having RUN the program

once, you should have all the names of the months in memory. Every time you want to use the program, use GOTO 490 to save having to retype the names of the months every time.

SUPPRESSING THE ERROR REPORT CODES

When you have a program where the display is very important (eg at an exhibition) or educational programs, it can be detracting or even embarrassing trying to explain "those funny little numbers at the bottom of the screen". Funny or not, here is a method whereby you can prevent the error report code from appearing.

The error report code is determined by the value of address 16384, the first system variable. The trick is to POKE numbers into 16384 that do not cause anything to be printed or to print spaces which, of course, cannot be seen. These values may be POKEd into 16384 for this purpose:- 43, 70, 72, 73, 74, 75, 76, 77, 79, 81, 82, 89.

Here is an example –

```
POKE 16384,74
```

You may find that certain numbers do not produce the desired result with certain programs. In this case, choose another number from the list. SAVE the program on tape before RUNning it if you're at all worried.

GREAT SYSTEM CRASHES

Careless use of POKE can ruin programs by overwriting vital parts or even cause a system crash where the computer appears to seize up and nothing you do will make it do anything except switch off.

Here are some of the exciting things you can do to your computer if you do like abusing it.

(A) Overwrite some of the NEWLINE characters, particularly in the display file – try this program:

```
10   LET P=PEEK 16396 +
     256* PEEK 1 6397
20   POKE P,0
```

Now try to get a normal display. The screen appears to have gone haywire if you press NEWLINE after running the program. All it does is find the start of the display file in line 10 from the system variable 16396/16397 which has this specific purpose, and changes the character found normally at this location (a NEWLINE character CHR$ 118) to a space by using POKE. The poor machine then gets confused when trying to produce a listing.

(B) For some novel displays, try POKEing all the numbers from 0 to 255 into the system variable 16384 that controls the error code.

(C) Load your favourite program, add a line or two to reset the frame counter to zero, and wait for a while to see the result (the frame counter is system variable 16436/7 and is reset to zero by POKE 16436,0 and POKE 16437,0). It may not work every time, but is usually quite effective.

(D) Try this program,

```
10   POKE 16418,0
20   INPUT A$
```

Where did the program go?

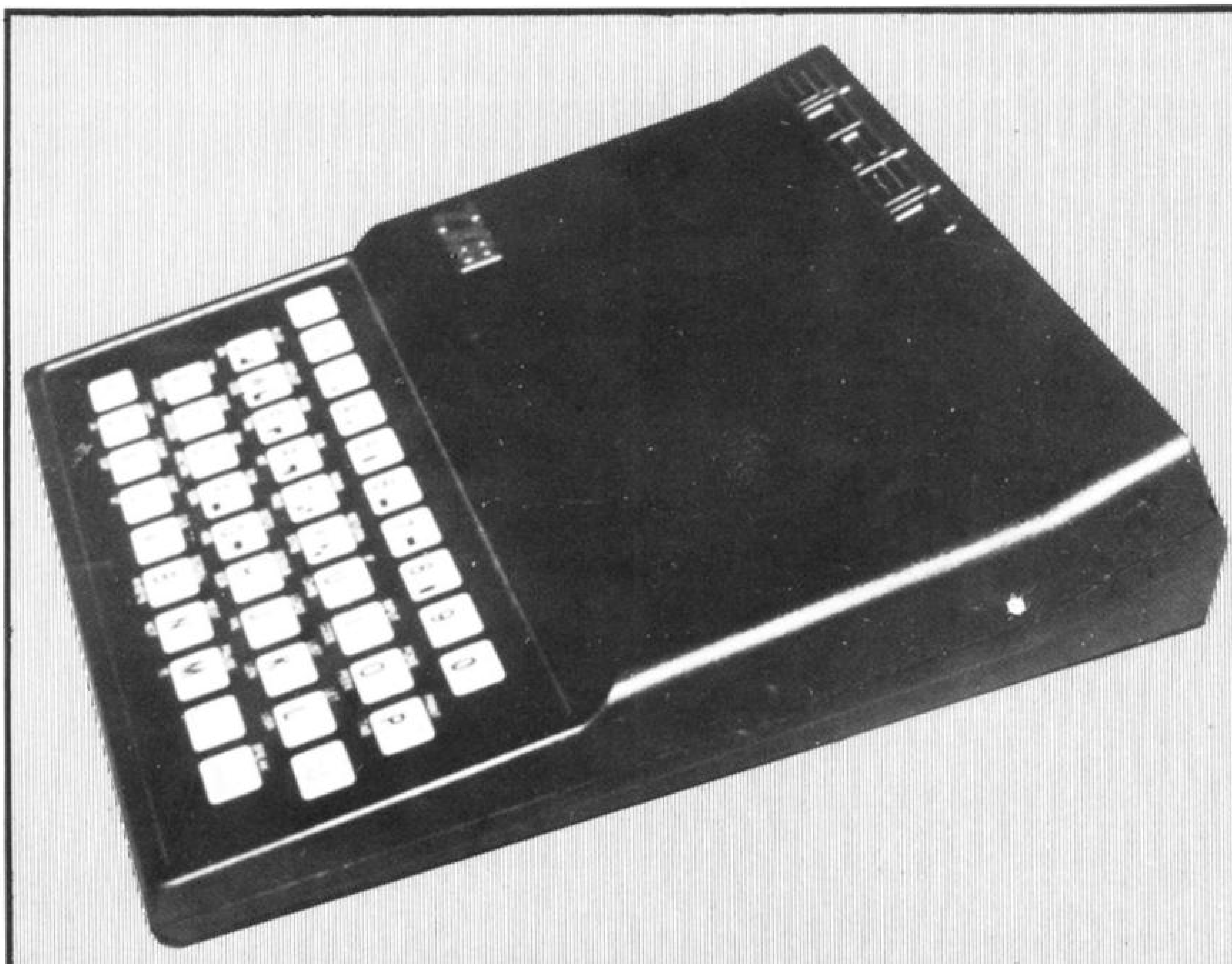
(E) This is the classic POKE anything anywhere at random. RUN it several times to see the different effects possible.

```
10   POKE 16384+INT
     (RND*1024),I NT
     (RND*1024)
20   GOTO 10
```

You may like to use the printer if you have one to keep a record of the interesting ones:

```
10   LET ADDRESS = 16384
     + INT (RND*1024)
20   LET R=INT (RND*256)
30   LPRINT "ADDRESS=";
     ADDRESS
40   LPRINT "NUMBER TO
     POKE =" R
50   POKE ADDRESS,R
60   GOTO 10
```

Too much fun gets boring. Back to some more serious things.



LENGTH OF PROGRAMS

Here is how the computer's RAM is organised:

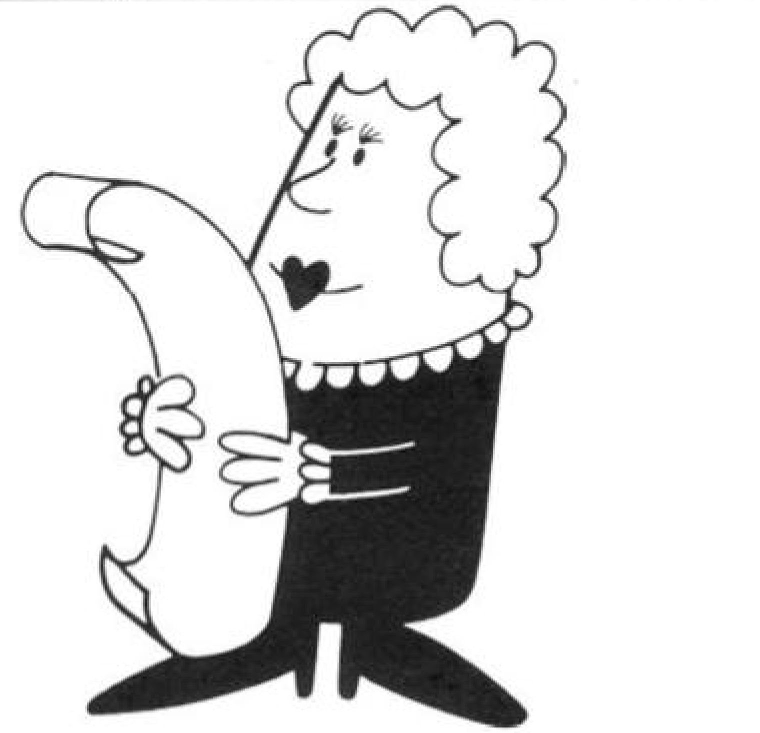
(i) system variables: 125 bytes

(ii) program alone excluding system variables, screen etc.
PRINT PEEK 16396 + 256 * PEEK 16397 – 16509

(iii) program, variables, system variables and display:
PRINT PEEK 16404 + 256 * PEEK 16405 – 16384

INSERTING NON-EDITABLE LINES INTO LISTINGS

(iv) memory left for user. This does not take into account the machine stack because the stack pointer cannot be accessed from BASIC:
PRINT PEEK 16386 + 256 * PEEK 16387 – PEEK 16412 – 256 * PEEK 16413 – 81
(it is necessary to subtract 81 because that is the length of the statement).



PART OF RAM	HOW TO FIND THE ADDRESS OF THE BOUNDARIES
	16384
SYSTEM VARIABLES	
	16509
PROGRAM	
	PEEK 16396 + 256 * 16397
DISPLAY FILE (SCREEN PICTURE)	
	PEEK 16400 + 256 * PEEK 16401
VARIABLES	
BYTE WITH CHRS 128	
	PEEK 16404 + 256 * PEEK 16405
WORK SPACE	
	PEEK 16410 + 256 * PEEK 16411
CALCULATOR STACK	
	PEEK 16412 + 256 * PEEK 16413
SPACE MEMORY	
	STACK POINTER - NOT ACCESSIBLE FROM BASIC
MACHINE STACK	
	PEEK 16386 + 256 * PEEK 16387
GOSUB STACK	
	PEEK 16388 + 256 * 16389

Normally, if you had a title/author REM statement in a listing, it is fairly simple to delete them, eg
1 REM (C) FRED BLOGGS 1982
10 . . . . . . (rest of program)
It is a simple matter to erase these lines using the EDIT facility or by typing in the line number. One method we can use is to change the line number of the first program line to 0. We know that the first line of a program starts at 16509, so since the line number is stored as the first two bytes of a line, we can use POKE to change these two bytes. Remember the two bytes are stored in the order MORE SIGNIFICANT BYTE followed by the LESS SIGNIFICANT BYTE (ie as you would write it — highest part first then the lowest part).

Here is how to change the line number to 0.

```
POKE 16509,0
POKE 16510,0
```

Now try to delete the first line. Quite secure, isn't it! The only way is to POKE a non-zero line number into 16509,10. So anybody who knew about the technique could easily delete the line.

A slightly better method is to change a line number in the middle of a listing. This is more difficult because we have no way of knowing where individual lines start. A starting point is the knowledge that program lines end with a NEWLINE character (CHR$ 118) and the next line will begin with the line number. Take this example:-

```
10  REM VAT CALCULATOR
20  PRINT "ENTER AMOUNT
    LESS VAT:";
30  INPUT A
40  REM (C) FRED BLOGGS
    1982
50  PRINT A
60  PRINT "VAT=";A*15/
    100
```

We need to change line 40 to line 0 and keep it located in its present position in the listing to make it difficult to delete or edit. Using the information we have, add these lines to the program:-

```
8000■FOR F=16509 TO PEEK
16396+2
56*PEEK 16397-3
9010 IF PEEK F=118 AND
256*PEEK
(F+1)+PEEK (F+2)=40 THEN
GOTO 90
40
9020 NEXT F
9030 STOP
9040 POKE F+1,0
9050 POKE F+2,0
```

Now delete lines 9000 to 9050 and then try to delete line 0!

Incidentally, it is normally better to insert this new line 0 at a point higher in a listing than line 255, since it will then be necessary to change 2 bytes of the listing to get rid of line 0, just to make it a bit safer. Another way to do the same thing is to use the system variable NXTLIN (16425/16426) to find the address of the start of the next line, provided you have space to add a few extra lines to the listing. We'll use this example:-

```
10  REM PATTERNS
20  INPUT A$
30  PRINT A$;
40  REM (C) FRED BLOGGS
    1982
50  GOTO 30
```

Add these extra lines to the program:

```
  39 LET A=PEEK 16425+
     256*PEEK 1
6426
  41 POKE A,0
  42 POKE A+1,0
  43 STOP
```

Now use RUN 39 to make the routine work. Once line 40 has been changed to line 0, delete the extra line. Incidentally, if you like making fools of computers, you can have great fun POKEing all sorts of line numbers into listings. Who said the computer sorts lines into order automatically?

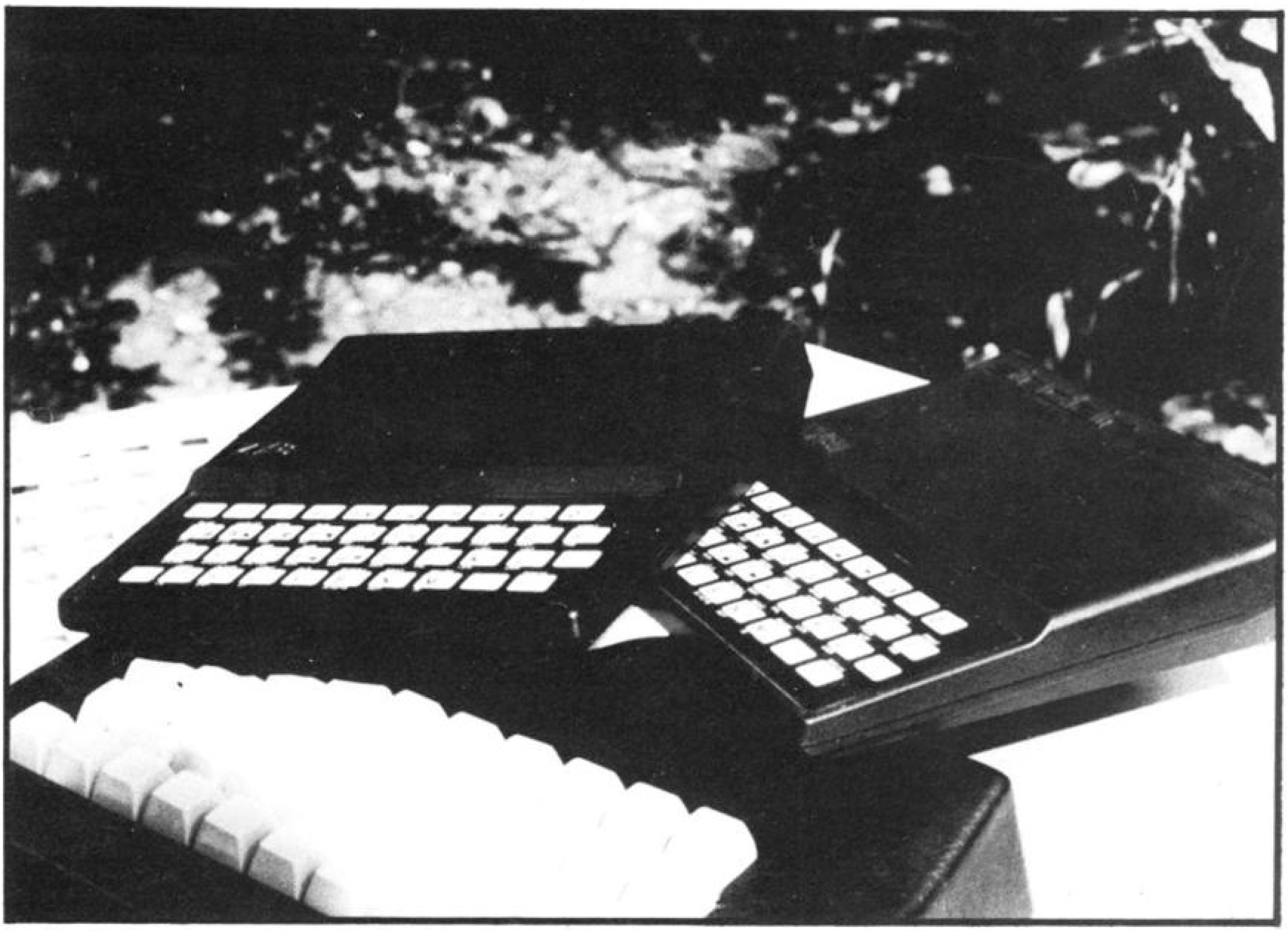
PREVENTING A SCREEN MEMORY OVERFLOW

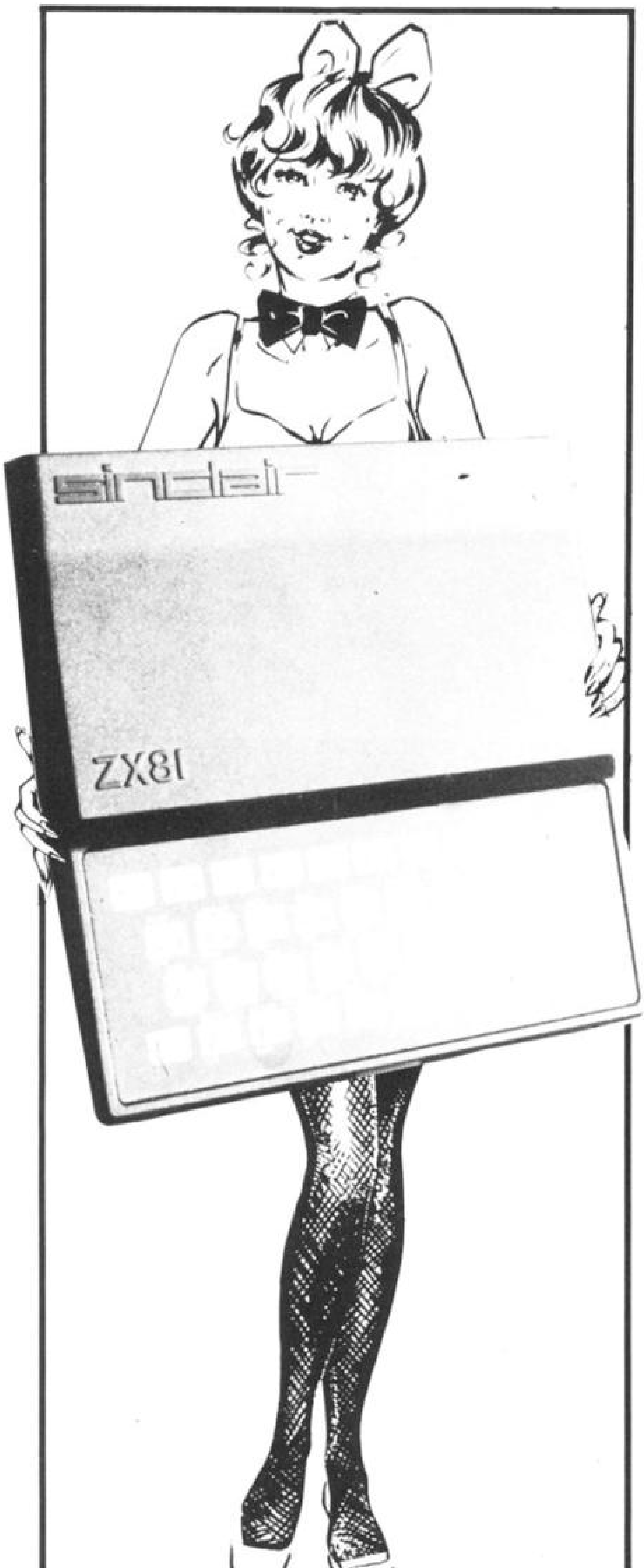
This routine makes use of the system variable 16442 which refers to the line number of the PRINT position, but does not have the same value as the line number. It starts off at 24 for the top line of the screen, down to 1 for the bottom line. The expression

IF PEEK 16442<4 THEN CLS

So if the PRINT position moves onto line 21 (the lowest line the user can PRINT on) the screen is cleared automatically.

Some programs require that the screen be cleared occasionally to prevent a screen memory overflow when the PRINT position gets down to the bottom of the screen. Here is one way to do this:-







IF PEEK 16442 < 4 THEN CLS
16442 is the system variable containing the line number of the PRINT position. It starts off at 24 for the top line, down to 3 for the lowest line available to the programmer and 2 and 1 for the two lines at the bottom of the screen used for INPUT etc. I have used 4, but you could substitute another number if you like.

Normally you can only PRINT on the top 22 lines of the screen display (lines 0 to 21). Any attempt to use the bottom two lines with PRINT is normally rewarded by an error report 5. You can gain access to these lines by two methods. The simplest is to POKE directly into memory at the location of the bottom two lines of the screen.

If you have more than 3¼K of memory plugged in (eg if you have a 16K RAMPACK) so that if the display is at full size, then line 22 starts at (PEEK 16396 + 256 * PEEK 16397 + 727), ends at (PEEK 16396 + 256 * PEEK 16397 + 758). Line 23 consequently starts at (PEEK 16396 + 256 * PEEK 16397 + 760) and ends at (PEEK 16396 + 256 * PEEK 16397 + 791). These addresses will be different if the display file size is altere, as might happen if SCROLL was used. The second method uses PRINT AT and the system variable DF — SZ at address 16418. The number in 16418 says how many lines in the bottom of the screen are not available to the user — normally two. So if we change this number to 0, we have access to all 24 lines of the screen display and we can use PRINT AT 23,X or PRINT AT 22,X.

However, this method comes unstuck when the computer tries to use the bottom of the screen for error reports, INPUTS, or even SCROLL. You can get a very nasty systems crash



and lose your program if you're unlucky (no lasting damage will be done, but you may have to switch off for a few seconds). The statement POKE 16418,0 must be entered as a line in a program.

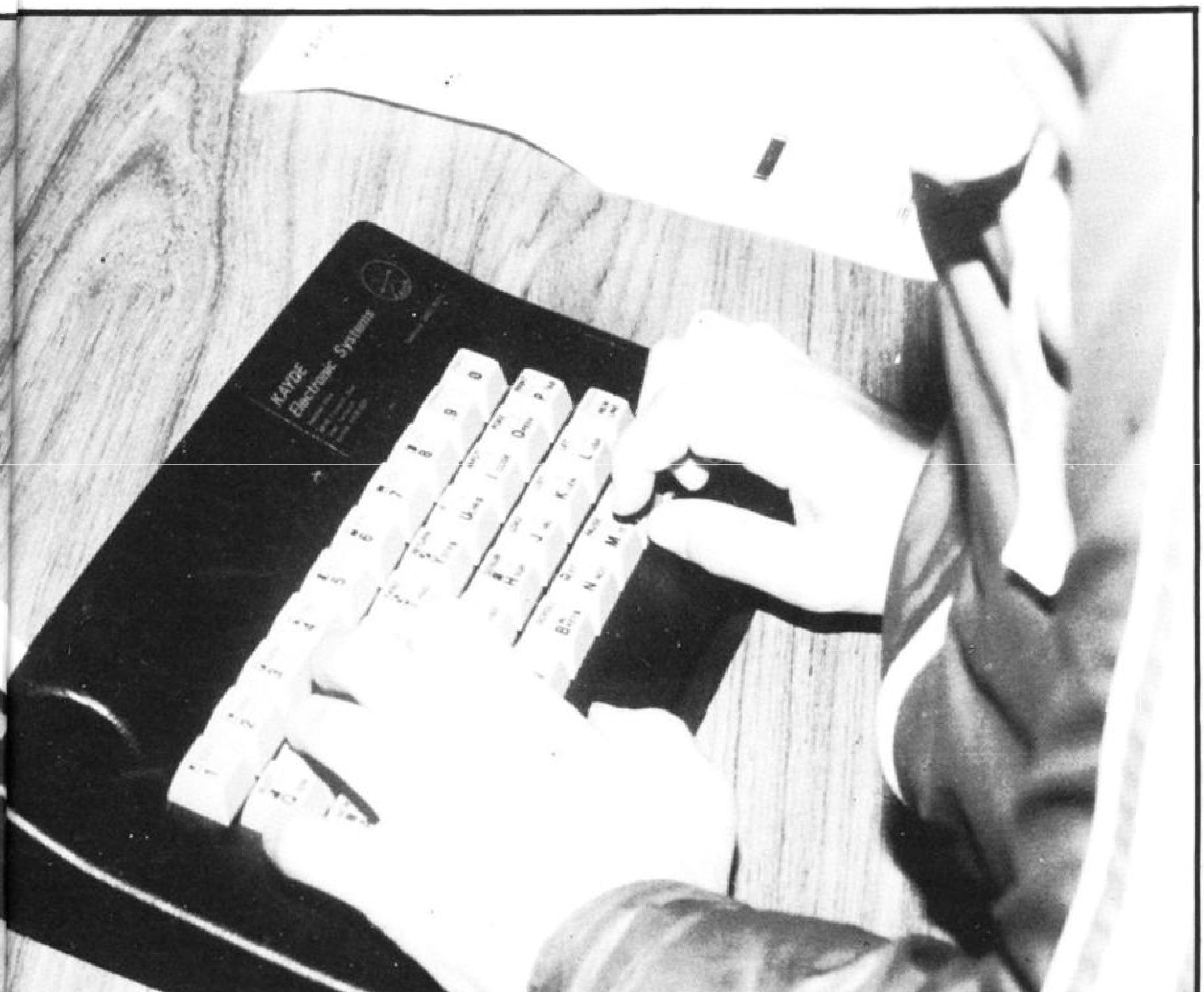
It does not work if entered as a direct command without a line number because the computer will reset it automatically when the screen is cleared, or a program is RUN. If you wish to use INPUT during the course of a program then you should POKE 16418,2 to restore the bottom of the screen to normal before attempting to use INPUT, which will of course erase characters PRINTed on line 22 and 23! Incidentally, be careful if you're using an unexpanded machine — the display file behaves in a strange way and makes use of 16418 so try not to upset it too much.

To place any particular line number you require at the top of automatic listings, you must first move the cursor to a line number greater than the one you want at the top. Then enter:-

POKE 16419,NUMBER — INT (NUMBER/256) * 256

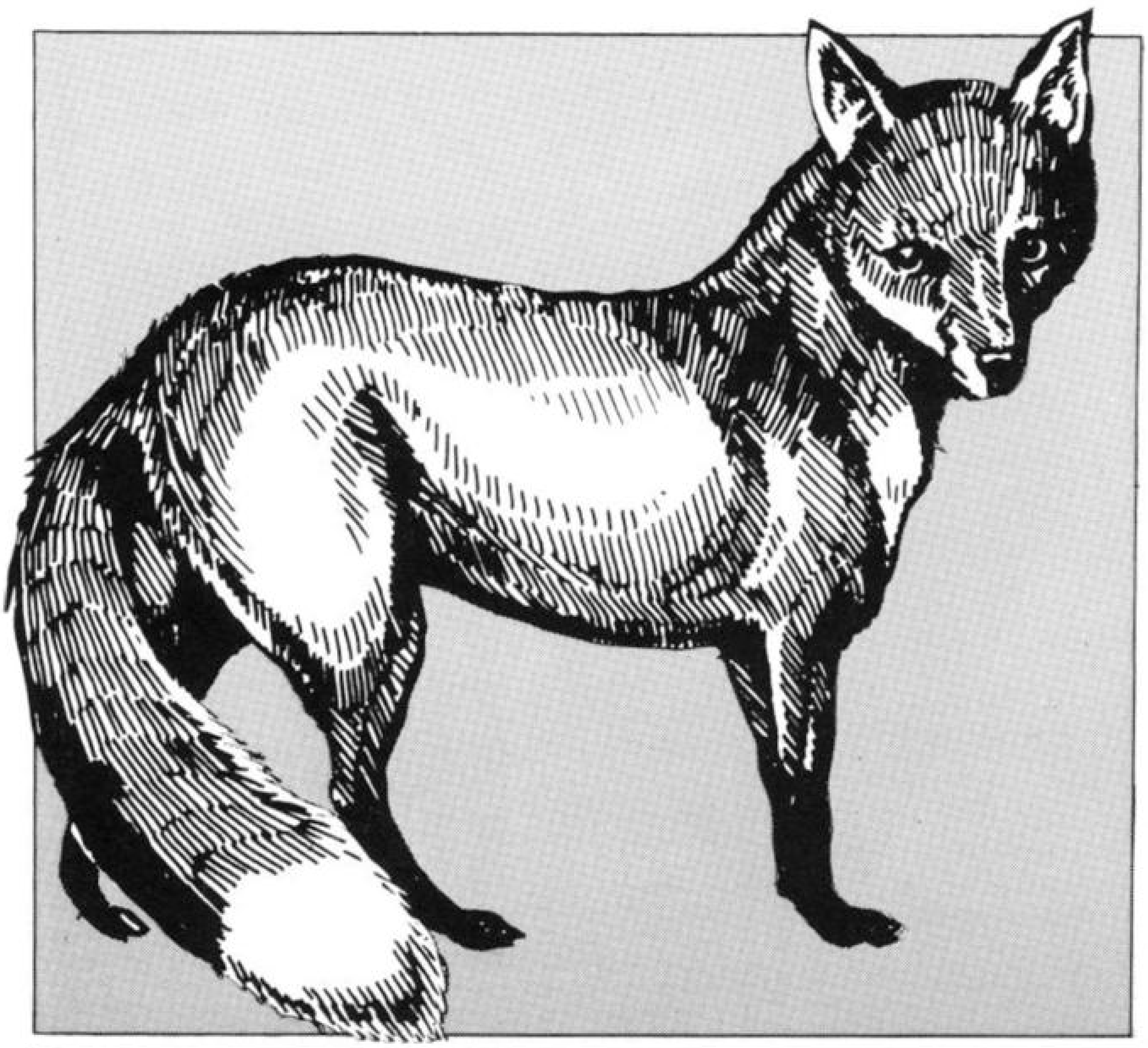
POKE 16420,INT(NUMBER/256)

Now when you press NEWLINE the automatic listing will begin where you specified (NUMBER is the line you want at the top of the screen). When entering lines when the cursor is at the bottom of the screen, the computer will usually compile the listing 2 or 3 times to get the new line onto the screen listing at the bottom. This is annoying, not to mention time-consuming. You can circumvent this like this. Type in any line number *higher* than any shown on screen and which does not exist in the listing (we always use 9999). The listing will change. If you now continue entering lines where you were originally, they appear near the top of the screen and the listing is made properly, saving a lot of frustration.



Foxing about

In FOX AND CHICKEN, written by Jim McCartney of Coleraine, you take part in a high speed chase through a maze. Playing the program calls on all your reflexes and luck — and demands an ability to keep cool in a crisis.



The object of this game is to get the chicken out of the maze without having it eaten by the fox. The program contains full instructions from line 100 down.

The program is a mixture of BASIC and machine code which would not suit any other machine. Most of the actual running of the game is in machine code; the BASIC is used to set it up and to draw the maze, etc. The BASIC is well enough annotated for a reasonably competent programmer to find his (her) way around it, but because the machine code can be tricky when it is put in the form of lines 76 and 77, I have given full assembly code details together with a description of the operations in code. Because bugs can easily creep in during the process of transferring coding from my typescript (guaranteed double checked) to your ZX81, check lines 76 and 77 carefully against both the BASIC listing and the machine code listing. If in doubt, check the machine code listing against the assembler codes in the left column, using Appendix A in the ZX81 handbook. (You will not find the assembler codes for Call KSCAN or Call FINDCHR; these refer to subroutines in the ROM). When you have done all this and everything agrees, SAVE the program before you RUN it! If it crashes, try checking the code again.

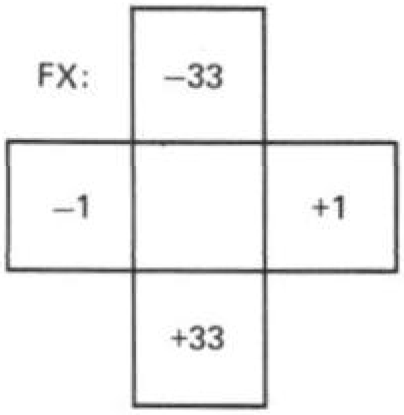


FIGURE 1a

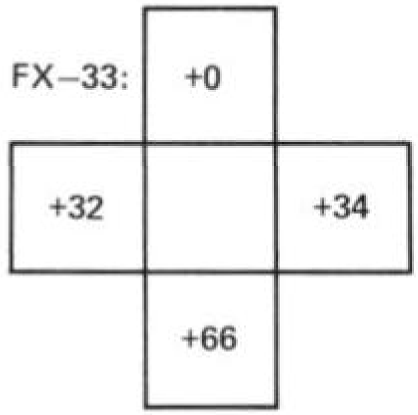


FIGURE 1b

How it works

The maze is drawn from line 1200 down, and it resides in the Display file. The address of the byte preceding the Display file is DFILE (line 1218). Each successive line in the display is numbered in the Display file as 33 more than the line above it, that is 32 display characters and a carriage return.

Either the Fox or the Chicken can move to any square next to it, provided that the Code of the character displayed in that square is 0; that is, provided that the square is empty. The Fox position is FX and the Chicken position is CK. To save switching between addition and subtraction in the machine code, the machine code references to the positions of these creatures are FX-33 and CK-33 respectively; you can see how this works from Fig. 1. Instead of using a system like Fig. 1a. we use the system in Fig. 1b. instead to explore the territory round FX and CK.

The fox moves automatically, but the chicken responds only to the keys. Each fox move is followed by an opportunity for the chicken to move, which it may or may not do depending on whether an appropriate key has been touched. There are four outcomes:

No key depressed	(RETURN) C = 20	go to next fox move, line 21
Fox catches Chicken	(CATCH) C = 40	go to line 40
Chicken escaped	(ESCAPE) C = 50	go to line 50
Chicken flies	(JUMP) C = 60	go to line 60

To find its way through all branches of the maze, the fox must turn consistently either to the left or to the right, during each run. This is set up at random in the BASIC program in lines 1680 down. If the fox is, eg, right-turning, the program will

a/ determine the direction in which the fox last moved

b/ explore the square to the right of this direction, and go there if possible

c/ ELSE explore the square ahead of this direction, and go there if possible

d/ Else . . . to the left

e/ Else . . . backwards

At least one of these must be available. The fox thus moves to the first vacant square in this sequence. For a left-turning fox, the sequence is left – front – right – back.

The miserable RND function on the ZX81 needed to be augmented in lines 60-62; other ZX81 enthusiasts who are irritated by RND may find this procedure helpful.

When you have the program up and running and debugged, you can delete lines 76 to 88 before you save the final version. The machine code will now be residing in the REM statement in line 1; you can see this when you LIST. Deleting 76 to 88 will speed the loading a bit and give a slightly faster startup.

BASIC LISTING

```
1    REM 12345678901234567890123456789012345
     67890123456789012345678901234567890l234
     56789012345678901234567890123456789012З
     12345678901234567890123456789012345678 9
     01234567890123456789012345678901234567 8
     9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
2    GOTO 75
5    REM * * * * * * * * RUN THE GAME
10   SLOW
12   PRINT AT 21,0;"RUN NUMBER";MZ
15   FOR J=1 TO 50
16   NEXT J
20   GOTO S
21   LET C=USR 16526
30   GOTO C
40   LET CK=33+PEEK 16524+256 *PEEK 16525
41   PRINT AT 0,12;"YUM YUM"
42   FOR J=1 TO 20
43   GOSUB 9000
44   POKE CK,23
45   GOSUB 9000
46   POKE CK,8
47   NEXT J
48   PRINT AT 0,12;" BURP "
49   GOTO 2000
50   PRINT AT 0,12;"ESCAPED"
58   GOTO 2000
60   LET R=RND
61   LET R=R *H-INT (R *H)
62   LET CK=DFILE+2 *INT (R *15)+6 6 *INT
     (RND *10)+35
63   POKE CK,23
65   LET JP=1
66   GOSUB 1700
67   LET JPNO=JPNO+1
68   IF JPNO=F THEN POKE 16622,99
70   GOTO 21
72   REM * * * * * * * * LOAD M/C CODE
75   FAST
76   LET  A$="2000002200204222002000003A8240
     218640BE28032318FA2B444D2A83400AC5F060
     0097EFE002808FE17287CC10318EA3608E5C600
     ED4206000E21093600E1C600ED42228340C10A3
     28240CDBB02444D51143E00284DCDBD077EFE21
     2812FE222812FE232812FE242812FE2B2851183
     33E20180A3E4218063E0018023E22A8C40E54F0
     600097EFE02807FE18282E118113617O6000E21C
     6 0 0 E D 4 2 2 8 C 4 0 E 1 0 9 3 6 0 0 "
77   LET A$=A$+"06000E14C9C600ED4206000E2109
     3600C106000E28C9E106000E32C92A8C4006000
     E 2 1 0 9 3 6 0 0 0 E 3 C C 9 "
83   LET X=16514
84   IF A$="" THEN GOTO 90
85   POKE X, 16 *CODE A$+CODE A$(2)-476
86   LET X=X+1
87   LET A$=A$(3 TO)
88   GOTO 84
90   DIM Z(4)
91   LET Z(1)=-33
92   LET Z(2)=-1
93   LET Z(3)=33
94   LET Z(4)=1
95   DIM T(7)
96   DIM J(4)
97   DIM M(150)
99   RAND
100  REM * * * * * * * * INSTRUCTIONS
105  SLOW
110  CLS
115  PRINT "DO YOU WANT THE INSTRUCTIONS? (Y/N)"
120  PRINT IF INKEY$ "Y" AND INKEY$ "N" THEN GOTO
     120
125  IF INKEY$="N" THEN GOTO 200
130  CLS
135  PRINT "THIS CHICKEN * LIVES IN A MAZE WHERE
     IT THINKS IT IS SAFE FROM THE FOX. . ."
137  PRINT
140  PRINT "BUT SOMEBODY HAS LEFT THE DOOR OPEN
     AND THE FOX         HAS GOT IN."
143  PRINT
145  PRINT "THE CHICKEN/S ONLY CHANCE NOW IS TO
     SLIP OUT WHEN THE FOX IS  LOOKING SOMEWHERE
     ELSE, BUT THE FOX IS VERY FAST AND THE
     CHICKEN IS NOT SO CLEVER."
147  PRINT
150  PRINT "YOU CAN GET THE CHICKEN OUT BY
     GUIDING IT WITH THE ARROWS ON KEYS 5 TO 9. IF
     THE CHICKEN IS IN GRAVE DANGER IT CAN FLY A
     SHORT DISTANCE INSIDE THE MAZE, BUT IT COULD
     LAND ANYWHERE."
153  PRINT
```

```
155  PRINT "HIT NEWLINE FOR MORE."
157  INPUT U$
'60  CLS
165  PRINT "THE CHICKEN CAN FLY ONLY A FEW TIMES
     IN EACH RUN. YOU CAN MAKE IT FLY BY HITTING
     F."
167  PRINT
170  PRINT "YOU GET TEN RUNS IN EACH MAZE AND
     THEN A NEW MAZE IS DRAWN. ALL THE MAZES ARE
     DIFFERENT."
173  PRINT
```

Comments

1 Leave 210 characters after REM to put the machine code in.

15 . . or use PAUSE

20 A very short delay

21 Run the game in machine code.

40 If C = 40 the fox eats the chicken.

50 If C = 50 the chicken escapes

60 If C = 60 the chicken flies.

65 JP is a flag used to RETURN in SUB 1700

67 JPNO counts the number of times the chicken flies.

68 POKE disables "F" key.

76 It is easiest to copy this off the machine code listing given separately. A$ can be broken into as many sections (as at line 77) as you find convenient for entry.

Once you have got all this RUN without bugs, you can delete lines 76 to 88 before you save the final version. The machine code wil now be safe in the REM statement in line 1. You can see this if you LIST.

85 Loads the machine code in A$ into the REM statement.

90 Sets up arrays. Z is used to draw the maze.

95 T is used to draw the maze.

97 M is the series of maze nodes in the order drawn.

190 Amend these instructions and the corresponding lines below to suit yourself.

220 S must be a line which says GOTO 21. The further down the listing it is, the slower will be the game.

1205 To watch the maze being drawn, put in 1206 SLOW.

1212 31 reverse spaces.

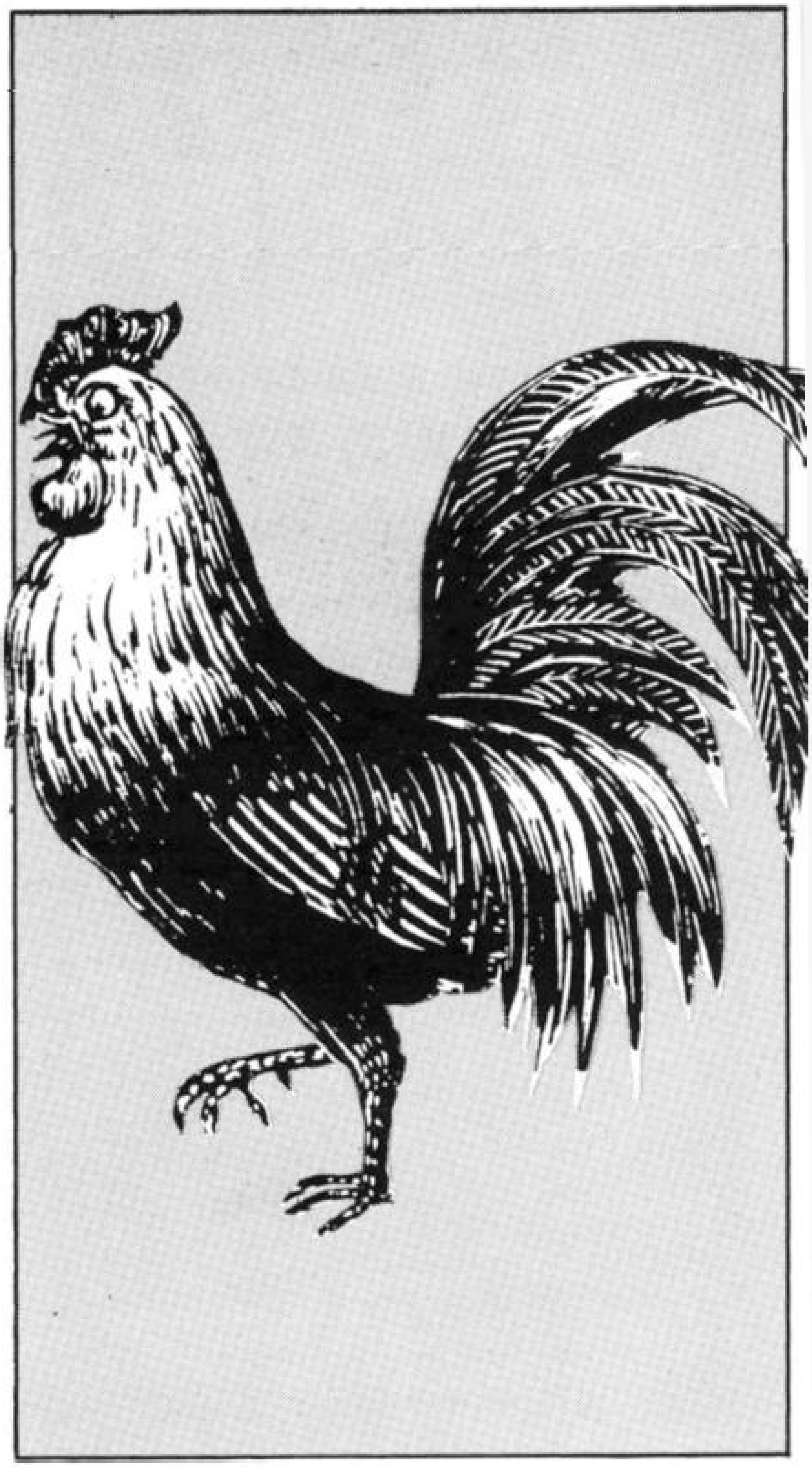
1220 Set up a random starting point for the maze.

1310 Explore the four directions possible from the current node of the maze.

1350 If it is possible to move in more than one direction in drawing the next node of the maze, a random possible direction is selected and drawn.

1360 If only one new direction is available, this is drawn. If no new direction, then backtrack.

1400 Go back through the maze as drawn, using the M array, until a node is found where a new direction can be drawn. If no such node is found, (M = 1), then maze is complete.



1510 Find an open space on the left side of the maze, and put the chicken in it.

1570 Find an open space on the right side of the maze, and put the fox in it.

1620 Draw an open space leading out of the maze beside the fox, and a door outside.

1650 Put the fox position in the machine code variables.

1680 Select a direction for the fox to turn.

1684 Change the machine code reference array accordingly.

1700 Put the chicken position in the machine code variables.

2000 Reset flag

2002 . . . or PAUSE

2010 Clear the fox and chicken positions.

2020 Put them back at the starting postions.



2040	Repair the top line of the maze.
2050	Add the score.
2065	Re-enable the "F" key
2067	Reset the flight counter
2070	Increment the run counter
2080	If 10 runs completed, finish.
3100	Start again.

MACHINE CODE VARIABLES

Address:
Hex Dec

Contents (initial):
Hex Dec

For:

Address Hex	Address Dec	Contents Hex	Contents Dec	For
82 4 0	16514	20	32	"Q" = direction of last fox move
83 4 0	16515	00	0	"FX – 33" = address of square
84 4 0	16516	00	0	above fox position.
85 4 0	16517	22	34	REFERENCE ARRAY for relative
86 4 0	16518	00	0	directions of fox move, set up (arbitrarily)
87 4 0	16519	20	32	for a right-turning fox.
88 4 0	16520	42	66	"
89 4 0	16521	22	34	"
8A 4 0	16522	00	0	"
8B 4 0	16523	20	32	"
8C 4 0	16524	00	0	"CK – 33" = address of square
I8D 4 0	16525	00	0	above chicken position.
8E 4 0	16526			start of program.

In addition to these, the machine code program is changed during the BASIC program at section 21; the line CMP 47,JRZ 73 (FE 2F 28 49) is replaced by NOP NOP NOP NOP (00 00 00 00) to disable the JUMP, after a preset number of jumps have been executed.

MACHINE CODE LISTING

			MOVE THE FOX
1	LDA,(16514)	3A 82 40	The number "Q" corresponding to the last fox move.
2	LDHL,16518	21 86 40	The first address to be examined in the reference array.
3	CP(HL)	BE	
4	JRZ,03	28 03	If A = datun in HL, then JUMP to the next stage at 6.
5	INC HL JR, –6	23 18 FA	Else increment HL and repeat till successful.
6	DEC HL	2B	Decrement HL (because we start looking to the right or left of the last fox move) and put it in BC.
7	LDB,H:LDC,L	44 4D	
8	LDHL(16515)	2A 83 40	The address of the square above the fox (FX – 33).
9	LDA,(BC) PUSH BC	0A C5	Get the contents the reference address from 7 and stack the address.
10	LDC,A LDB,0 ADD HL,BC	4F 06 00 09	Put the *contents* back into BC and add to HL. This is the address of the next square to be searched adjacent to the fox.
11	LDA,(HL) CPO JRZ,8	7E FE 00 28 08	Search it. If zero, there is an open space; jump to 14.
12	CP23 JRZ,124	FE 17 28 7C	If 23, there is a chicken: goto CATCH.
13	POP BC INC BC JR, – 22	C1 03 18 EA	ELSE get the old reference back, increment it and repeat from 8 till successful.
14	LD(HL),8 PUSH HL	36 08 E5	Poke 8 to the new fox position and stack the position.
15	ADD A,0 SBC HL,BC LDB,0 LDC,33 ADD HL,BC LD(HL),0	C6 00 ED 42 06 00 0E 21 09 36 00	Change HL back to the old fox address (FX – 33); add 33 to get the true address and poke 0 to it.
16	POP HL ADD A,0 SBC HL,BC LD(16515),HL	E1 C6 00 ED 42 22 83 40	Get the new fox address back, subtract 33 to get the new "FX – 33" and put it in 16515/6 for the next time round.
17	POP BC LDA,(BC) LD(16514),A	C1 0A 32 82 40	Put the contents of the reference address into 16514 to give the new "Q" (see 1)
21	LDA,32:JR 10 LDA,66:JR 8 LDA,0 :JR 2 LDA,34	3E 20 18 0A 3E 42 18 06 3E 00 18 02 3E 22	Depending on the result of 20, put the appropriate MOVE difference in A (Fig. 1b).
22	LDHL(16524) PUSH HL LDC,A:LDB,0 ADD HL,BC	2A 8C 40 E5 4F 06 00 09	Get the address of the square "CK – 33" above the chicken and stack it. Put the MOVE in BC and add to the above address to get the TEST address.
23	LDA,(HL) CP 0:JRZ 7 POP HL:JR 17	7E FE 00 28 07 E1 18 11	Search the TEST address. If it contains 0, continue; if 24, goto ESCAPE; else clear the stack and goto RETURN.
24	LD(HL),23	36 17	Poke 23 to the TEST address for the new chicken position.
25	LDB,0:LDC,33 ADD A,0:SBC HL,BC LD(16524),HL	06 00 0E 21 C6 00 ED 42 22 8C 40	BC = 33; reset carry and let HL – 33 be new "CK – 33".
26	POP HL ADD HL,BC LD(HL),0	E1 09 36 00	Get back the old "CK – 33"; add 33 to give CK and poke 0 to blank the old chicken position.
			RETURN
27	LDB,0:LDC,20 RET	06 00 0E 14 C9	The game continues; put the line number (20) in BC and return.
			CATCH
28	ADD A,0:SBC HL,BC LDB,0:LDC,33 ADD HL,BC LD(HL),0:POP BC LDB,0:LDC,40 RET	C6 00 ED 42 06 00 0E 21 09 36 00 C1 06 00 0E 28 C9	As 15; fox moves onto chicken. Then clear stack, put 40 in BC and return.
			ESCAPE
29	POP HL LDB,0:LDC,50 RET	E1 06 00 0E 32 C9	Clear the stack. Put 50 in BC and return.
			JUMP
30	LD HL(16524) LDB,0:LDC,33 ADD HL, BC LD(HL),0 LDC,60 RET	2A 8C 40 06 00 0E 21 09 36 00 0E 3C C9	Get "CK – 33"; add 33 and blank CK with 0. Put 60 in BC and return.

Adding a keyboard to the Spectrum

Stephen Adams looks at the electronics beneath the ZX keyboards, and tells you how to add an extension keyboard of your own.

The three ZX computers produced by Sinclair – the ZX80, ZX81 and the ZX Spectrum – all use the same type of keyboard. The method of getting information into the three machines is also the same, so I will only go over the Spectrum Keyboard pointing out the differences between the various machines.

The keyboard itself is made up of three layers of plastic. The top layer is coated on the inside with metal strips in a grid pattern which go horizontally under five keys. Each set of five keys has a different metal strip running under it. For example, keys 1 – 5 have one metal strip running under all the keys. The bottom plastic sheet has a grid of vertical metal lines running under four keys.

This metal grid runs under two sets of keys one on each side of the keyboard. Keys X-D-E-3 are connected to the same strip as M-K-I-8. The middle sheet only contains a set of forty holes, one under each key, so that when a key is pressed the upper metal strip can touch the lower metal strip. This forms an electrical contact switch between the two grids. Which wire of top grid is connected to which wire of the bottom grid is determined by the key pressed.

By putting voltage on each of the eight top wires in the metal grid and by testing each one of the wires on the bottom grid we can tell which key has been pressed. Every key has a unique combination of one top grid wire (address line) and one bottom grid wire (data line).

The main difference between the ZX80/81 and the Spectrum is that the keys on the Spectrum have been covered with a flexible rubber sheet which has key tops moulded on to it.

This sheet is suspended over the keys so that it flexes when a key top is pushed down giving a much needed feel to the keys which the ZX80/81 did not have.

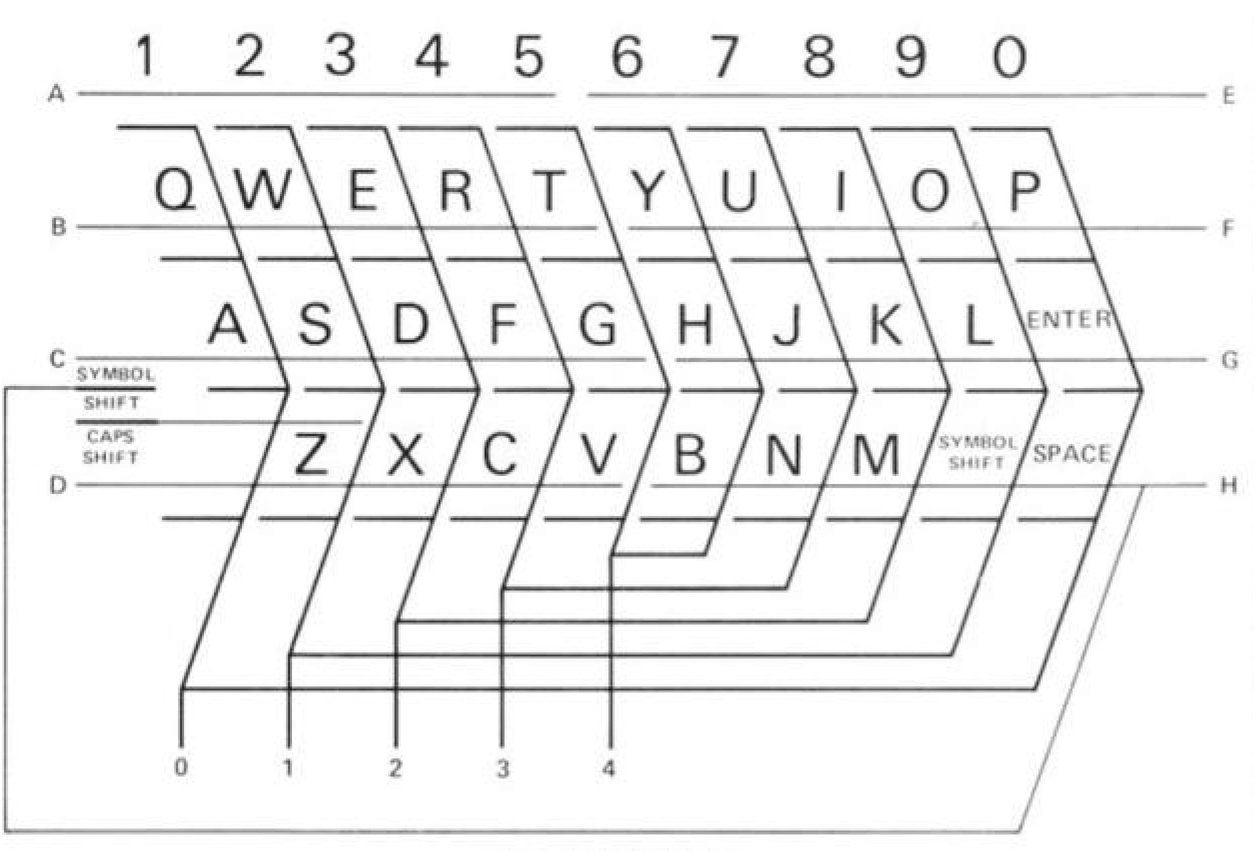
Such is the importance of having 'feel' on the keyboard of a ZX computer (which can increase the speed at which information can be typed in by up to 50%) that many people have fitted an extra keyboard to replace the one that Sinclair provides.

This extra keyboard is usually made up of forty keyswitches which are laid out in the same pattern as Sinclair's keyboard. Each keyswitch has wires which are joined together only when the key has been pressed down, so they act just like the keys on the Sinclair keyboard.

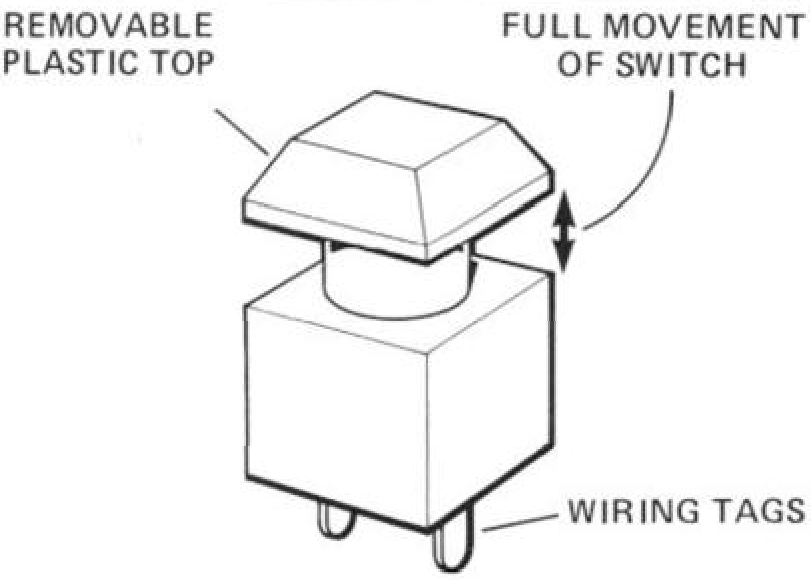
The difference is that these keys can be pushed down up to half an inch and can therefore be felt moving under the fingers. This movement or 'feel' allows you to release the key as soon as it has reached the bottom of the key's movement and go on to the next key without having to check whether the key was registering on the computer.

It was very difficult to do this on the ZX80/81 as the downwards movement was only 0.1 inch and consequently many people kept their fingers longer on the keys than necessary.

With the ZX Spectrum this has improved, but I think that many people would still like a proper moving keyboard for two reasons. One, so that the keys

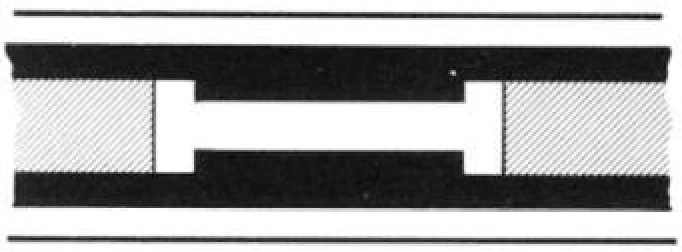


THE ZX KEYBOARD
FIGURE 1



A TYPICAL KEY SWITCH

FIGURE 2



SINCLAIR SWITCH

FIGURE 3

can be felt, but not every key press is registered on the computer. The second is that the ZX Spectrum now has two SHIFT keys called Symbol Shift (which is the second key in from the right hand side on the bottom line) and Caps Shift key (on the far left hand side of the bottom line). With Sinclair's keyword system of entering commands both keys are in constant use, either separately or together and this makes it very awkward to use as the user is constantly crossing his or her hands switching from one shift key to another. The move of the symbol shift key next to the caps shift key on a new keyboard would make it a lot easier to use as both keys could be pressed with one hand while the other searched for the appropriate key.

Constructing a keyboard

There are four ways of constructing a ZX keyboard:—

1. Build it up from a kit purchased from a component manufacturer.
2. Obtain an old computer keyboard and strip away all the coding (usually a printed circuit board) so that the keys can be rewired like Sinclair's.
3. Make up a printed circuit board and purchase some key-switches for it, the printed circuit board holding the keys in place.
4. Make up a box from metal sheet and mount the keys on it. All that is required then is to wire it up from key to key using the same matrix as the Sinclair Keyboard.

Assuming you want to do 2 or 4 a wiring diagram is required (such as shown in fig. 1). The keys are wired up five at a time horizontally, dividing the keyboard into eight lines.

Each line will have a wire attached to it which will go to a particular address line on the computer's keyboard socket; it is important that these lines are wired up correctly as although it will cause no damage to the computer if wrongly connected, it will not give the correct or in some cases any response. This completed, the data wires can be wired in to the other side of the switch. All the switches should be of the push-to make single pole type and should only have two tags. The data wires run vertically up the keyboard and it is better if this is done starting from the end of the keyboard and working inwards.

Each data line (there are only five) is taken to two lines of keys, the outermost keys are connected together (Caps Shift-A-Q-1-0-P-ENTER-SPACE).

ZX80 and ZX81 users will have SHIFT instead of CAPS SHIFT and NEWLINE instead of ENTER on their keyboards. The next vertical line of keys on the far right hand side is connected to the next line of keys on the left hand side. This ends up with the two middle vertical rows being joined together.

On the ZX Spectrum, the *symbol shift key can either be* moved to the position shown in Fig. 1 or an extra key provided there. Both should be wired as shown.

The best keys to use are those with clear plastic tops as the labels for the keys can be put underneath them. If you haven't got any covers then paint out the tops of the keys with white paint and using various coloured inks, write the functions on the keys. Spectrum users will have quite a bit to do as there can be up to 6 functions to each key.

Connecting it up to the computer.

To make it easy to identify the connections of the 13 wires used on each computer, I have labelled the wires A-H for the address wires and 0-4 for the data *wires. See* Fig. 4 *for the* connections to your computer.

All the connections should be made to the underside of the keyboard sockets or in the case of the ZX80 directly to the printed circuit board as this allows you to use the original keyboard as well.

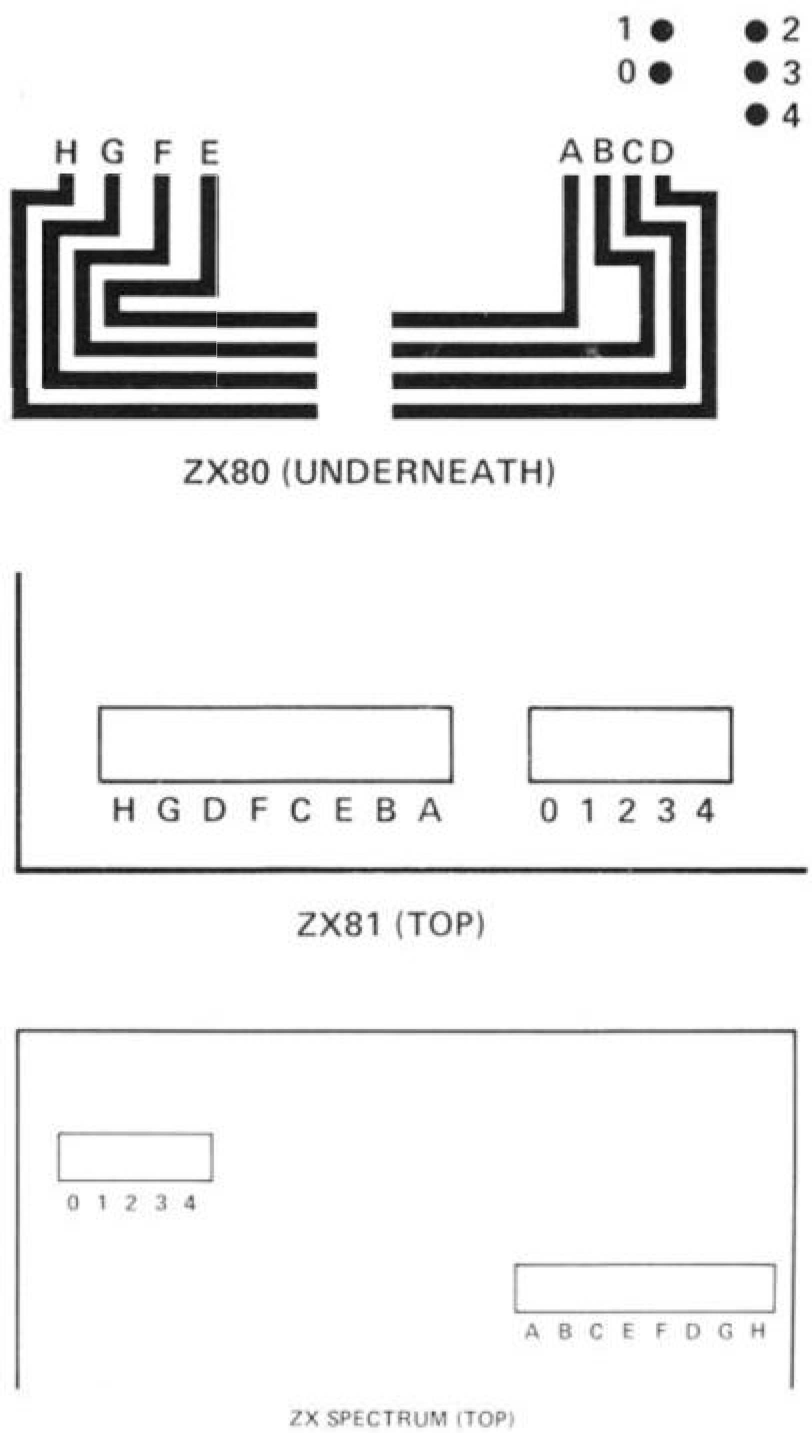


FIGURE 4

Slot mac

Catch one-armed bandit fever with thi 7K program for the 16K ZX81 written b Adam Waring and Mike Cleverley of Hul

SLOT MACHINE uses a flashy machine code routine to reverse the display. It is called during the introduction, winning and losing routines.

The object of this game is to win a grand total of £50. This is achieved by gambling on the one-armed bandit (see lines 40 to 80). It costs £1 per go, and you win £5 for getting two numbers the same, £15 for getting three the same.

The program has RESPIN and NUDGE routines.

This is the routine to put the machine code into the REM statement.

```
   1 REM 12345678901234567890
  10 INPUT X
  20 LET A$=""
  30 IF A$="" THEN INPUT A$
  40 IF A$="S" THEN STOP
  50 POKE X,16*CODE A$+CODE A$(2
) -476
  60 LET X=X+1
  70 LET A$=A$(3 TO )
  80 GOTO 30
```

After you've entered that, run it and input the following:

```
2A, 0C, 40, 06,  17, 2B, 23, 7E,
FE, 76, 20, 03,  10, F8, C9, C6,
80, 77, 18, F2
```

Line 1 should now look like this:

```
   1 REM E£RND▖▝*F7 SAVE TAN LEN
▓?/ PAUSE
```

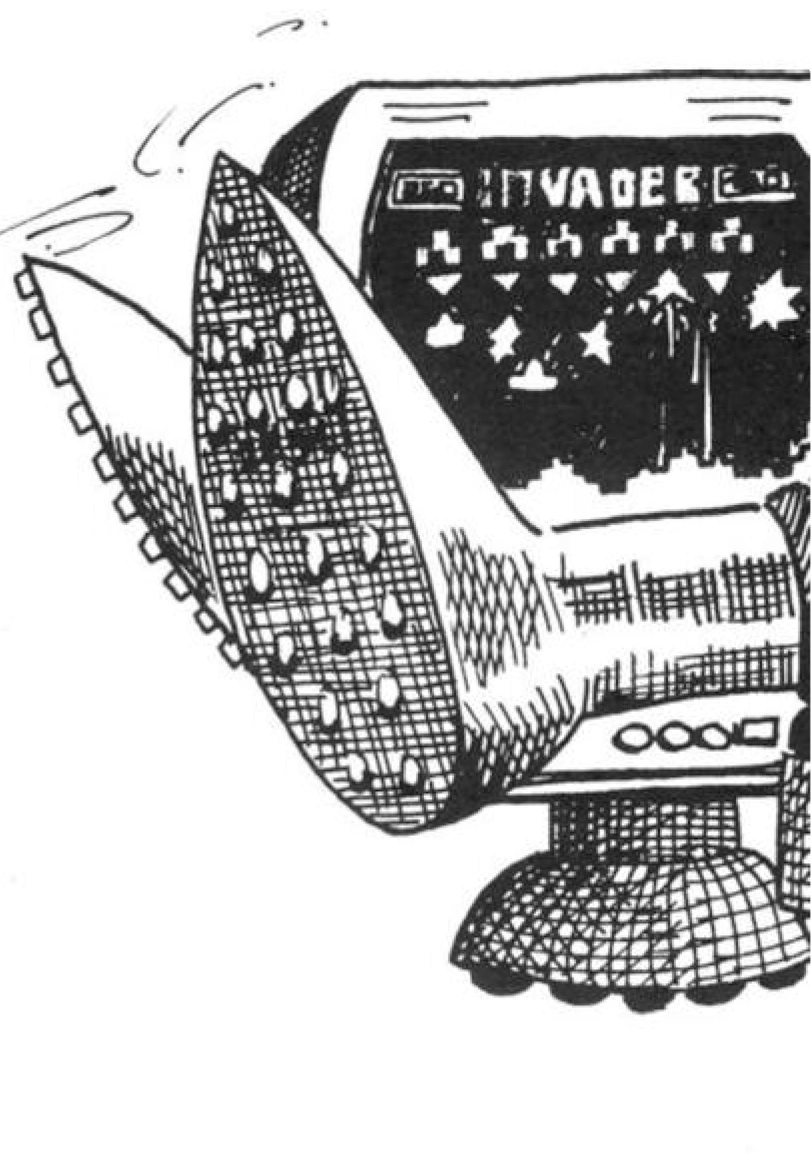
Once it does, enter as a direct command POKE 16510,0

Then, enter the rest of the program:

```
   0▓REM E£RND▖▝*F7 SAVE TAN LEN
▓?/ PAUSE
   2 REM
   3 REM      SLOT MACHINE
   5 REM    COPYRIGHT    (C)
   6 REM   MIKE CLEVEPLE
   7 REM         AND
   8 REM     ADAM WARING
   9 REM
  20 GOSUB 2000
  21 FOR N=1 TO 21
  22 SCROLL
  23 NEXT N
  24 LET P=PEEK 16396+PEEK 16397
*256
  25 RAND
  27 DIM A(4)
  30 LET A=5
```

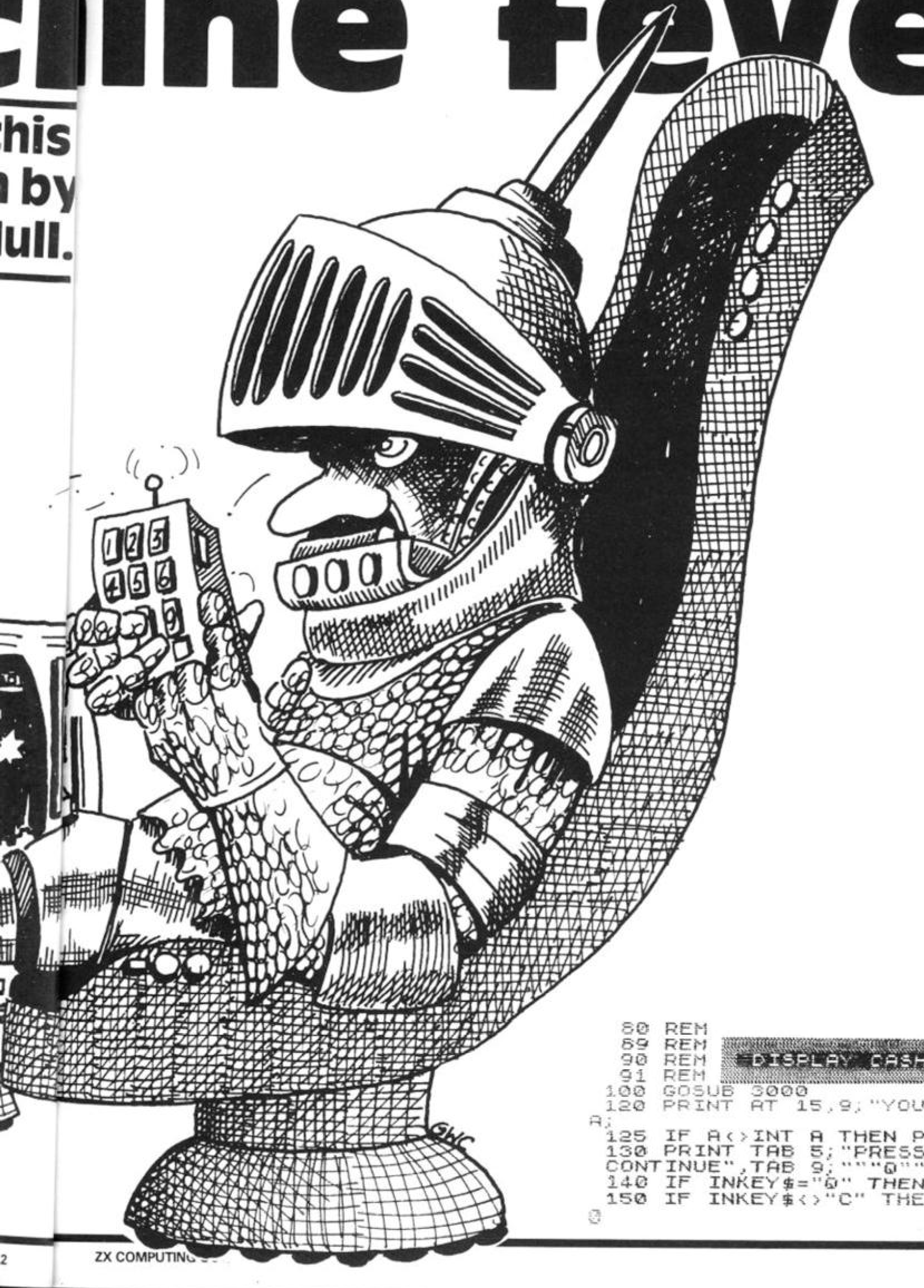
```
  34 REM
  35 REM   PRINT MACHINE
  36 REM
  38 CLS
  40 PRINT
  43 PRINT
  46 PRINT
  50 PRINT
  53 PRINT
  56 PRINT
  60 PRINT
  63 PRINT
  70 PRINT
  73 PRINT
  76 PRINT
  78 PRINT
  79 PRINT
```





chine fever

his
by
ull.



```
 80 REM
 89 REM
 90 REM  DISPLAY CASH
 91 REM
100 GOSUB 3000
120 PRINT AT 15,9;"YOU HAVE £";
A;
125 IF A<>INT A THEN PRINT 0
130 PRINT TAB 5;"PRESS ""C"" TO
CONTINUE",TAB 9;"""Q"" TO QUIT"
140 IF INKEY$="Q" THEN STOP
150 IF INKEY$<>"C" THEN GOTO 14
0
```

```
180 GOSUB 3000
181 REM
182 REM NUMBERS FOR REELS
183 REM
185 FOR L=1 TO 3
190 FOR M=1 TO 10-L*2
195 FOR N=L TO 3
200 LET A(N)=INT (RND*10)
203 POKE P+166+N*2-1,A(N)+28
205 NEXT N
206 NEXT M
207 NEXT L
208 REM
209 REM RESPIN OR NUDGE?
210 REM
211 LET G=RND
212 IF G<.7 THEN GOTO 4000
215 IF G<.8 THEN GOTO 400
240 REM
250 REM RESPIN
260 REM
270 LET X=INT (RND*3)*2+1
303 PRINT AT 16,0;"DO YOU WANT
A RESPIN?"
307 PRINT AT 8,X;"█"
311 PRINT AT 8,X;"▣"
313 IF INKEY$="N" THEN GOTO 400
0
315 IF INKEY$<>"Y" THEN GOTO 30
7
320 FOR N=1 TO 20
324 LET A((X+1)/2)=INT (RND*10)
326 POKE P+166+X,A((X+1)/2)+28
328 NEXT N
350 GOTO 4000
400 REM
401 REM NUDGE
402 REM
420 LET G=INT (RND*4+2)
430 PRINT AT 15,3,"YOU HAVE ";G
;" NUDGES"
450 FOR N=1 TO G
460 IF INKEY$="0" THEN GOTO 400
0
470 IF INKEY$<"1" OR INKEY$>"3"
THEN GOTO 460
480 LET B=CODE INKEY$-28
490 LET A(B)=A(B)-1
```

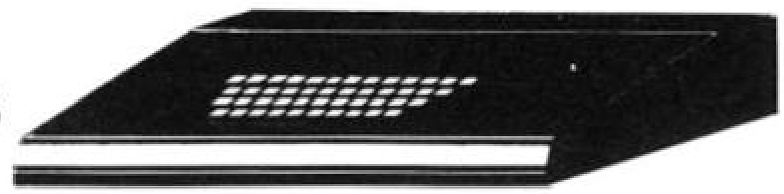


```
495 IF A(B)<0 THEN LET A(B)=9
500 PRINT AT 5,1;A(1);TAB 3;A(2
);TAB 5;A(3)
510 NEXT N
520 GOTO 4000
2010 REM
2011 REM INTRODUCTION
2012 REM
2050 PRINT
2100 PRINT "[illegible]"
2110 PRINT "[illegible]"
2120 PRINT "[illegible]"
2130 PRINT "[illegible]"
2140 PRINT
2150 PRINT "[illegible]"
2160 PRINT "[illegible]"
2170 PRINT "[illegible]"
2180 PRINT "[illegible]"
2185 LET B$="MIKE CLEVERLEY AND
ADAM WARING"
2190 LET A$="COMPLETED ON 19TH J
UNE 1982 BY"
2205 FOR N=1 TO 30
2210 PRINT AT 11,N;A$(N)
2215 PRINT AT 12,31-N;B$(31-N)
2220 FOR M=1 TO 5
2230 NEXT M
2240 NEXT N
2273 FOR N=1 TO 21
2275 RAND USR 16514
2276 FOR M=1 TO 7
2277 NEXT M
2278 NEXT N
2280 LET A$="
INSTRUCTIONS?"
2294 PRINT AT 14,0;A$
2300 LET A$=A$(2 TO )+A$(1)
2305 IF INKEY$="N" THEN RETURN
2310 IF INKEY$<>"Y" THEN GOTO 22
90
2311 REM
2312 REM INSTRUCTIONS
2313 REM
2320 CLS
2330 PRINT "INSTRUCTIONS"
2340 PRINT "------------"
2360 PRINT "TRY YOUR LUCK AT THE
 ONE ARMED  BANDIT. WIN A TOTAL
OF £50."
2370 PRINT "YOU ARE GIVEN £5 TO
START OFF"
2380 PRINT "WITH. EACH SPIN COST
S £1. YOU   SPIN BY PRESSING ""C
""."
2390 PRINT "GETTING 2 REELS THE
SAME WINS   YOU £5. GETTING 3 TH
E SAME WINS"
2400 PRINT "£15. DURING THE GAME
 YOU MAY GETA RESPIN. THESE ARE
FREE, AND"
2410 PRINT "THE FLASHING BUTTON
INDICATES   WHICH REEL MAY BE RE
SPUN. YOU"
2420 PRINT "RESPIN BY PRESSING T
HE ""Y"" KEY, IF YOU DO NOT WISH
 TO RESPIN"
2430 PRINT "THEN PRESS THE ""N""
 KEY."
2431 PRINT
2432 PRINT
2433 PRINT
2435 PRINT "PRESS ""C"" TO CONTI
NUE"
2436 IF INKEY$<>"C" THEN GOTO 24
36
2437 CLS
2440 PRINT "NUDGES ARE ALSO AVAI
LABLE AT    RANDOM STAGES THROUG
HOUT THE"
```

```
2450 PRINT "GAME. YOU WILL HAVE
FROM 2 TO 5 NUDGES AT A TIME. YO
U PRESS THE"
2460 PRINT "COLUMN NUMBER, ""1""
, ""2"" OR ""3"", TO NUDGE THE A
PROPIATE COLUMN."
2470 PRINT "TO STOP NUDGING, PRE
SS ""0""."
2480 PRINT "THE GAME MAY BE TERM
INATED AT  ANY TIME BY PRESSING
""Q"" OR "" ""."
2483 PRINT
2484 PRINT
2485 PRINT
2490 PRINT "PRESS ""C"" TO CONTI
NUE"
2495 IF INKEY$<>"C" THEN GOTO 24
95
2500 RETURN
3000 REM
3002 REM  CLEAR SCREEN
3006 REM
3020 FOR N=13 TO 21
3030 PRINT AT N,0;"
                              "
3060 NEXT N
3070 RETURN
4000 REM
4002 REM  MONEY WON AND LOST
4004 REM
4010 GOSUB 3000
4050 LET A=A-1
4060 LET A(4)=A(1)
4100 FOR N=1 TO 3
4110 IF A(N)=A(N+1) THEN GOSUB 4
500
4120 NEXT N
4400 IF A>50 THEN GOTO 5000
4410 IF A<1 THEN GOTO 6000
4490 GOTO 80
4500 FOR M=13 TO 21
4520 PRINT AT M,1;"00000"
4530 PRINT AT M-1,1;"     "
4570 NEXT M
4580 LET A=A+5
4590 RETURN
5000 REM
5002 REM  WOW, YOUVE WON
5004 REM
5009 CLS
5050 PRINT
5060 PRINT "
"
5070 PRINT "
"
5080 PRINT "
"
5090 PRINT
5100 PRINT "
"
5110 PRINT "
"
5120 PRINT "
"
5125 PRINT
5130 PRINT "
"
5140 PRINT "
"
5150 PRINT "
"
5160 PRINT
5170 PRINT "                 ""
Q"" TO QUIT"
5180 PRINT "                 ""
S"" TO START"
5190 PRINT "
"
5200 PRINT
5210 PRINT "
"
5220 PRINT "
"
5230 PRINT "
"
5300 RAND USR 16514
5304 FOR N=1 TO 10
5306 NEXT N
5310 IF INKEY$="S" THEN RUN 21
5320 IF INKEY$<>"Q" THEN GOTO 53
00
5330 PRINT TAB 8;"GOODBYE SUCKER
"
5500 STOP
6000 REM
6010 REM  YOURE A BORN LOSER
6020 REM
6030 CLS
6050 PRINT
6060 PRINT
6100 PRINT "
"
6110 PRINT "
"
6120 PRINT "
"
6130 PRINT "
"
6140 PRINT
6150 PRINT
6160 PRINT "
"
6170 PRINT "
"
6180 PRINT "
"
6190 PRINT "
"
6200 PRINT
6210 PRINT
6220 PRINT "
"
6230 PRINT "
"
6240 PRINT "
"
6250 PRINT
6260 PRINT
6270 PRINT "     ""S"" TO START",""
Q"" TO QUIT"
6300 GOTO 5300
6550 SAVE "BANDI[inverse T]"
6560 RUN
```



COBRA 1000
THE COMPLETE BUSINESS SYSTEM FOR THE ZX81



The Cobra computer system allows you to expand your Sinclair ZX81 into a powerful and efficient computer for the small business.

- Accommodates a full range of business hardware
- Capacity for up to 9-12 boards
- PCB slide guide systems ensure total stability
- All add-on hardware comes complete with leads and sockets — no soldering of drilling necessary
- Durable, lightweight metal computer case
- Fully professional keyboard — with space bar

HIGH QUALITY, FULLY BUILT BASIC MODEL £79.95 + VAT

Expand your Cobra with a comprehensive choice of add-ons:

• Interfaces • Tape drive unit • Upper and lower case character board • High resolution graphics • Input/output ports • Light pen • Colour Modulator.

THE CHEAPEST INTERFACES ON THE MARKET

RS 232 Interface

At last a high quality 232C interface to allow you to connect your ZX81 to any 232C printer from the cheapest dot matrix to the most expensive daisy wheel. Look at these features:

- Baud rate variable from 110 to 9600 with software control
- Upper and lower case printout
- Up to 120 haracters per line
- Small and compact — fits on to the back of your ZX like a RAM pack
- Uses the ZX power supply
- Output via a small 3 pin jack.

Price only £26 + VAT

Centronics Interface

Allows you to connect the ZX to any Centronics printer. Has similar features to the 232C interface — one exception is that output is via a 24 way socket at the back of the unit.

Price only £26 + VAT

Coming Soon: ZX81 SPECTRUM MICRODRIVE INTERFACE

Connect the Sinclair to the new Spectrum Microdrive! As soon as the microdrive is available we will be developing an interface for it to be connected to your ZX.

Interfaces will also be available to adapt this example of modern micro engineering for use with Apple and Pet computers.

This is just one of the future developments by Cobta Technology.

MICROLINE A PRINTER

• Print speed 120 characters per second • Bidirectional printing • 9 x 7 dot matrix • 5-10 and 6.5 characters per inch • 8 or 10 lines per inch • Vertical tabulation and form feed • Form length user selectable • Upper and lower case • Block graphics • 1 line print buffer • 80 characters per line

Price £281.75 inclusive of VAT

COBRA TECHNOLOGY LIMITED
378 Caledonian Road, Islington, London N1 1DY

Please send SAE for brochure

Microprocessors and bus systems

We look at the most commonly used microprocessors, and delve into the mysteries of the S-100 bus.

The most commonly used microprocessor chips in the hobby market are the 8080, Z-80, 2650, SC/MP and 6502. Other variants can be easily spotted — the 8085 is very similar to the 8080 but with certain changes. The ZX machines are built around the Z-80A chip, a development of the Z-80.

Which is the best one? This is a difficult question — it's like high-level languages (of which there are many different types and variants), people who are used to a particular one will prefer it to any other.

Long arguments develop between programmers over the good and bad points of each language. It's the same way with processors.

The 8080 is probably the processor with the most 'software support' — it has the most programs written for it. The Z-80 can run any program written for the 8080, as well as some which the 8080 cannot.

The SC/MP has the advantage that it needs practically no 'support chips' — it will more or less stand alone and is thus ideal for many 'dedicated' applications, such as doorbells, alarms, etc.

The major differences between the processors in terms of programming are the instruction sets and the number of registers.

The instruction set of a processor is a list of all of the different arithmetic and logical operations it can perform — like the number of keys on a calculator. The registers in the processors are the same as calculator memories — the more, the better.

The 8080 instruction set is about the same level of complexity as the 2650 and the 6502. This is adequate for most applications.

The SC/MP has a rather limited instruction set and relies on its ease of application for its appeal.

The Z-80 instruction set includes the 8080 set — and them some! It also has twice the then some! It also has twice the though, it is usually felt that the 8080 level of complexity is sufficient for the beginner.

S-100 and all that

What exactly is the S-100 bus? Or any bus for that matter? No, they're nothing to do with public transport. The word 'bus' is short for 'omnibus' (literally: 'for all'). Basically, it's a method of interconnecting parts of a computer system so that they can communicate with each other.

It takes the form of a 'backplane' or 'mother board' which holds several edge connectors. Printed circuit boards can be plugged into these, one edge of the board being covered in gold-plated strips right up to its edge. Contacts on the edge connector make electrical contact with these strips. The S-100 bus system uses double-sided boards with 50 strips per side (thus the 100 in S-100!).

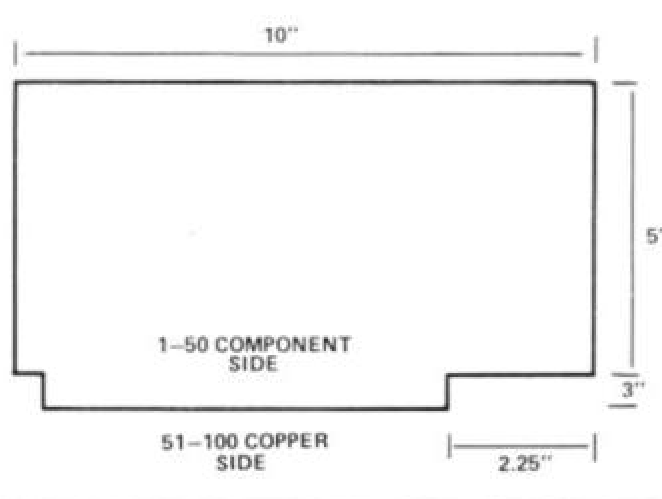
Each board — one of which will be the microprocessor board, holding the micro chip itself plus all the other 'support' chips necessary to get the thing to work, such as oscillators and buffers etc — has some outputs and some inputs which are connected to the bus in a standard configuration. There are sixteen lines of the bus which carry information on 'addresses'. This is how a position in memory is defined — by a sixteen digit binary number.

When the microprocessor wants to find out what's at a particular address on the board which carries the memory, it puts that address on the sixteen address lines, put out a request on some of the other lines of the bus and the memory board looks up the required information and puts it onto the 'data' section of the bus. The microprocessor board then reads the data from the bus.

Other buses have differing numbers of lines and the positions of the data and address lines are also different but they work in essentially the same manner. Unfortunately, it is difficult to connect a board intended for one bus system to a board intended for another.

For this reason, each manufacturer either uses his own bus structure, as is the case with Sinclair, or sees the light and uses the S-100, which is about as close to a standard as the hobby computer field has.

Fig. 1. Standard S100 cards are 10" by a nominal 5.3". Some manufacturers change the height depending on circuit requirements. Edge connector spacing is 0.125", offset to prevent backward insertion of a board.



Ready, steady, go!



In the first issue of ZX Computing, Henry Budgett and Tim Hartnell discussed the standard benchmark tests used to test the speed of various functions on microcomputers. Stephen Tyler and Mark Dulling of Kingsbridge in Devon decided to get out their stopwatches, and put a number of popular computers through their paces, to see how they measured up.

We were interested in the benchmark tests for testing the ZX81's speed in the Summer issue of ZX Computing, and have run the following tests, including 33 of our own additional tests on different types of computers.

We did all timings in minutes and seconds, to two decimal places, and used the maximum abbreviation possible on the micro being used (such as N. for NEXT on the Atom and the BBC Micro).



The computers we used were:

Sinclair ZX81	belonging to S. Tyler
Commodore PET	belonging to T. Mellor
BBC Microcomputer	all belonging
Acorn Atom	to Kingsbridge
Research Machines 380Z	Upper School.

The results for benchmark 1:

	SLOW	FAST
ZX81	17.79 Floating Point	4.58 Integer
PET	1.54	
BBC	0.49	0.18
ATOM	0.49 (approx.)	
380Z	1.1 (approx.)	

Bench mark 1 was re-run on the computers but this time with lines 1 to 9 consisting of:
REM 12345678901234567890
The speed of the ZX81 dropped to 20.88 in SLOW mode and 5.31 in FAST mode. The other computers were unaffected.

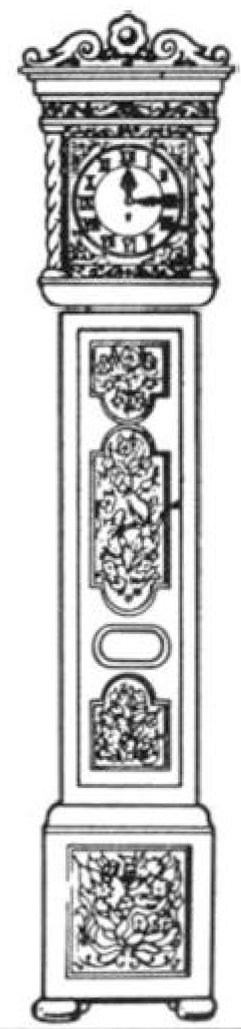
The results for the benchmarks are given below:-

	1	2	3	4	5	6	7	8
ZX81-SLOW	17.79	27.04	1:05.13	1:02.97	1:13.51	3:19.91	4:36.45	1:31.75
FAST	4.58	6.84	16.32	15.78	18.43	49.76	1:08.72	22.95
BBC Floating Point	0.49	2.74	7.77	8.23	8.53	12.37	19.55	4.59
Integer	0.18	1.66	7.29	7.43	7.73	9.45	13.71	4.54
PET Floating Point	1.54	9.60	18.00	20.00	21.50	32.00	50.00	9.62

The following 32 tests consist of a loop from 1 to 1000 with a different function inserted in the loop for each test. The ZX81 functions in a program line have been written in the left column; equivalents have been used in other computers. The times given are the times taken for the loop with the function with the time for the open loop subtracted to give the time of the function 1000 times.

They are arranged in the order of the fastest to the slowest in terms of the ZX81.

No.		SLOW-ZX81-	FAST	BBC Floating Pt.	Integer	PET Foating Pt.
1	PRINT "2";	4.43	1.26	0.99	0.96	—
2	LET B=2	6.06	1.48	0.77	0.42	1.84
3	CLS (1KZX81)	6.12	1.58	45.58	45.58	—
4	LET B=NOT 2	8.08	1.96	0.89	0.50	—
5	LET B=SGN 2	8.26	2.03	0.86	0.48	—
6	LET B=2 AND 3	8.87	2.16	1.05	0.70	—
7	LET B=2 OR 3	9.05	2.20	1.05	0.71	—
8	LET B=ABS-2	9.78	2.42	0.88	0.52	—
9	LET B=CODE "A"	10.31	2.52	0.69	0.44	1.81
10	LET B=LEN "A"	10.53	2.61	0.80	0.43	2.29
11	LET B=INT 2.5	10.99	2.70	1.91	1.56	6.65
12	LET B=PEEK 2	11.44	2.79	0.76	0.50	—
13	LET B=2<>3	11.58	2.85	1.09	0.70	—
14	PLOT 2,2	13.03	3.20	2.69	2.69	—
15	LET B$="2"	13.67	3.35	0.49	0.48	—
16	LET B$=INKEY$	14.11	3.50	0.81	0.81	1.81
17	LET B$=CHR$ 100	19.47	4.80	0.80	0.79	—
18	LET B=VAL "2"	27.18	6.73	0.90	0.55	2.16
19	LET B=RND	53.63	13.30	1.53	1.67	6.16
20	LET B=EXP 2	2:45.56	41.67	7.53	7.74	27.16
21	LET B=SIN 2	2:48.11	41.77	23.79	24.09	29.46
22	LET B=COS 2	2:54.51	43.27	23.26	23.22	29.06
23	LET B=ATN .5	4:00.17	59.97	5.34	5.29	44.46
24	LET B=LN 2	4:25.91	1:05.95	15.77	16.08	—
25	LET B=TAN 2	5:45.51	1:25.75	42.87	43.13	54.46
26	CLS (16KZX81)	6:05.61	1:30.53	45.58	45.58	—
27	LET B=SQR 2	6:47.89	1:41.16	10.03	10.31	—
28	PRINT 2;	7:17.49	1:48.86	12.89	12.88	—
29	LET B=2**2	7:26.66	1:50.77	4.01	4.23	—
30	LET B$=STR$ 100	7:35.21	1:53.11	6.93	6.92	—
31	LET B=ASN .5	11:57.89	2:57.92	33.04	33.35	—
32	LET B=ACS .5	12:03.01	2:59.32	33.28	33.55	—



The ZX81 in FAST mode is occasionally faster than the PET but is only faster than the BBC microcomputer at clearing the screen when it has less than 3¼K of RAM according to the system variable RAMTOP when the display file is not full.

The table (right) shows the average relative speeds of the computers with the BBC microcomputer using integers given a value of 1 and the rest given higher values because of their slower speeds.

BBC	(Integer)	1
BBC	(Floating Pt.)	1.22
PET	(Floating Pt.)	3.25
ZX81	(FAST)	4.03
ZX81	(SLOW)	16.20

COMPETITION

WIN A ZX PRINTER



Win a ZX printer, and a complete set of Psion/ZX81 software!

We're looking for six clever programmers — and we've got great prizes to give the winners: a ZX printer for first prize, and five sets of Psion software for the ZX81 for the runners-up.

All you have to do is write a clever program — in BASIC -which fits within 1K on the ZX81, and has something to do with trains! It doesn't matter if your program ties damsels in distress to disused tracks in Southern Region, or helps lost and weary travellers find their way on the Central Line of the Underground, or whatever . . . but the program must tie in, in some way, with trains.

Your entry must be as a clear listing (printer-dumped if you can convince someone to lend you one), or splendidly hand-written. No cassettes, please. This competition is not open to permanent contributors to this magazine, or any other of the computing periodicals. We'll be printing the best entries in the next issue of ZX Computing, when we'll also be announcing the lucky winners. The entered program must not have been previously published, must be your own original work, and must not have been submitted for publication to any periodical. Entry to the competition will be construed as permission to publish the program. No entries can be returned.

This competition closes with the last mail on October 11, 1982. No correspondence regarding this competition will be entered into, and — as they say in the classics — the judges decision will be final.

Send your entry to: ZX Printer Competition, ZX Computing, Argus Specialist Publications, 145 Charing Cross Road, London WC2.

Noisy little things



ZON X-81 sound unit

In this issue, our hardware reviewers look at a system which allows your ZX81 to listen, a ZX81 sound maker or three and a device to ensure that you don't lose a program when the power fails.

ZON Sound Unit

A wide range of sound effects can be added to your ZX-81 with the ZON X-81 Sound Unit now available from BI-PAK.

The unit is based on a three-channel-plus-noise sound chip, and is so designed that the pitches and volumes of the three channels and the overall attach/decay envelope can be controlled by simple BASIC statements. By this means, piano, organ, bells, helicopters, lasers and explosions can be simulated and easily added to existing programs.

ZON X-81 is completely self contained in a neat black plastic case with loudspeaker and manual volume control (in addition to programmed volume), and simply plugs in between the rear of the ZX-81 and its RAM pack and/or printer (if fitted). No dismantling, wiring, soldering, batteries, power supplies or leads are required.

Instructions take you through the operation of the unit step by step and include a number of example program of useful sound. It is available from BI-PAK Semiconductors, P.O. Box 6, Ware, Herts, WARE 3442/3182. Price £25.95, including postage and V.A.T.

Big Ears

The 'Big Ear's speech recognition system, which costs £49, plus VAT, including p&p, consists of a microphone, pre-amplifier, analogue frequency filters and digital interface, complete with software. Words are stored as voice patterns which the system learns from repetition by the user.

Ten or so words can be stored at a time.

The computer then checks each word spoken against its word bank, and assigns a percentage to each word regarding its chance of being the word spoken. It assumes the word gaining the highest score is the spoken word, and responds accordingly.

It is available from William Stuart Systems, Dower House, Herongate, Brentwood, Essex, CM13 3SD (0277-810244).

Protecting that program

The 'software protection unit', made by Microbyte of Lichfield,

looks like being a partial answer to all those horrifying moments when 8K worth of carefully typed in programs suddenly vanishes.

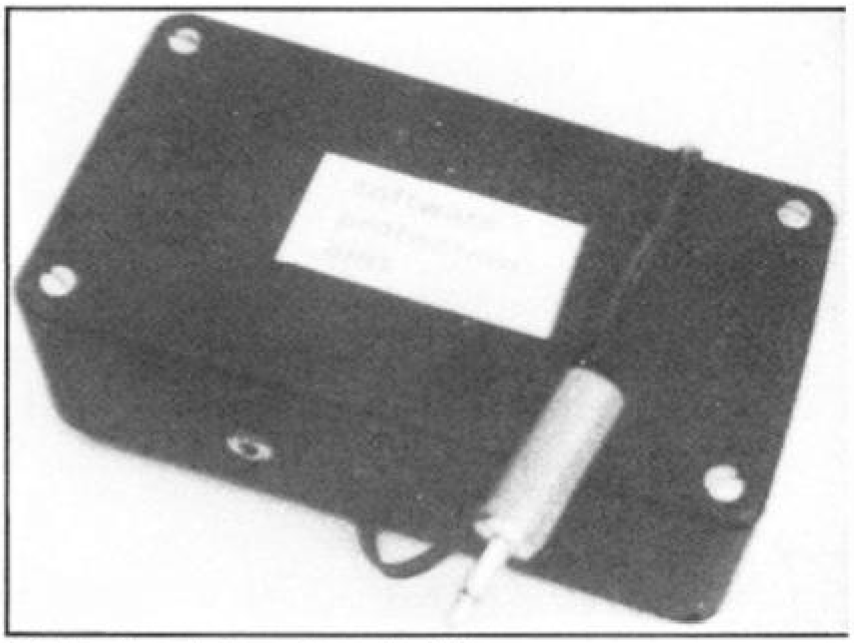
The software protection unit has a socket to take the mains power unit jack plug from your ZX81 mains transformer, and the plug on the unit then hooks into the ZX81.

If you unscrew the four screws on the lid of the unit, you'll see two battery containers, which hold a total of six batteries. These are the secret of the unit, which is designed to take over if the power fails.

When we tried it, we found it worked exactly as described. We didn't try the 'how long will a program last' test, but take the manufacturer's word that with a ZX81 alone, a program will stay intact for three hours, and with the 16K attached, about an hour and a half.

You're cautioned to always remove the power jack plug on the ZX81 after turning off the mains supply, or the batteries will think there has been a power supply, and wear themselves out keeping the '81 going. The batteries are, of course, only intended for emergency use. Excessively used batteries are signalled by video character deformation, and loss of sync, although at this stage the RAM contents are not lost.

The unit costs £8.60 and is available from Microbyte, 19 Worcester Close, Lichfield, Staffs, (05432) 28556.



The Microbyte Software Protection Unit

Making Music

If you want to add sound to your ZX81 the Bolton Electronics Music Chip may be of interest to you.

This is a GIAY-3-8910 sound generator chip interfaced to the ZX81. The chip is mounted on a printed circuit board with all the necessary interfacing components.

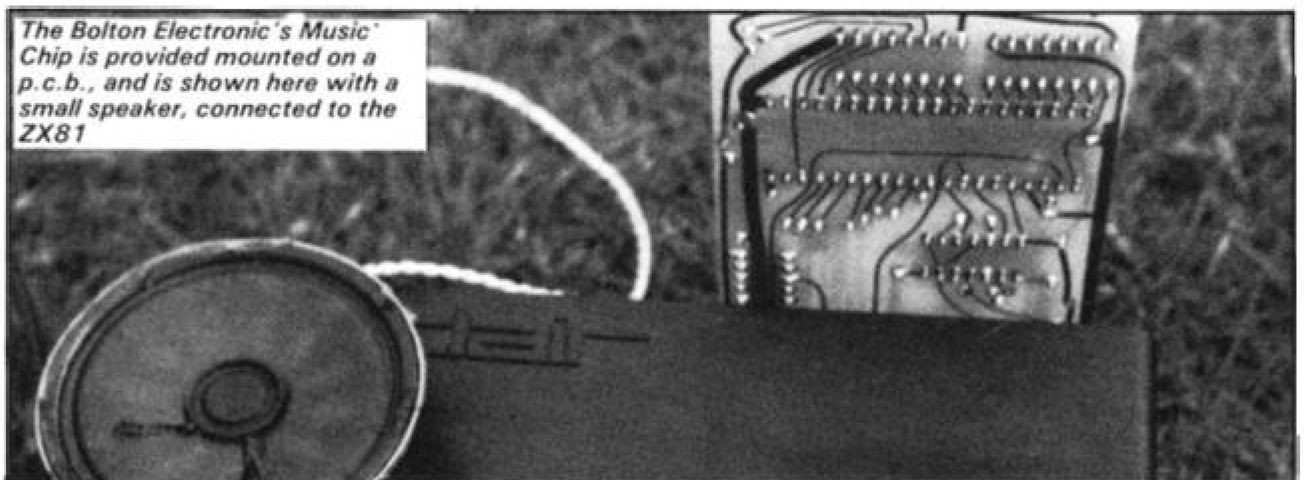
The p.c.b. plugs in to the rear of the computer.

The computer's edge conector is reproduced behind the p.c.b. to enable RAM packs and printers to be connected.

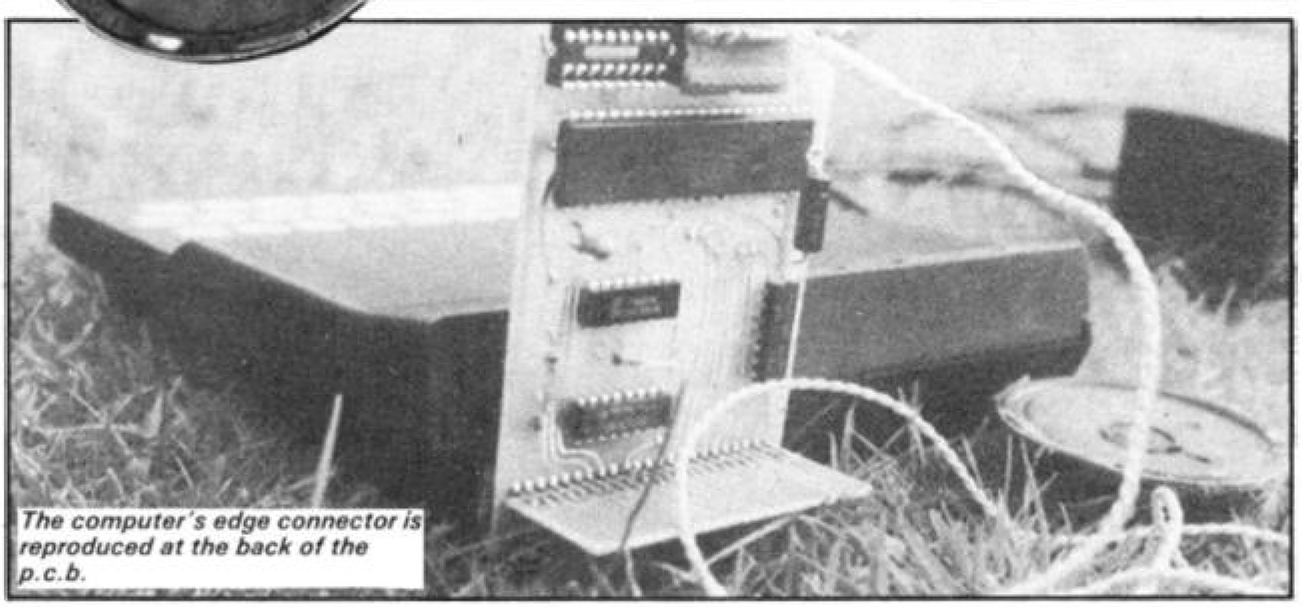
The music chip is a fairly tricky little device. It provides three separate audio channels and each of these can be programmed with separate frequencies and volumes.

A programmable noise generator can also be switched on to the three channels and a programmable envelope generator can be set up to control the volume of the three channels, so you can see there's a fair degree of flexibility built in to this particular product.

It also includes an on-board audio amplifier which has enough output for a small loudspeaker (which you don't get with the kit).



The Bolton Electronic's Music Chip is provided mounted on a p.c.b., and is shown here with a small speaker, connected to the ZX81



The computer's edge connector is reproduced at the back of the p.c.b.

The amplifier input and output connections are brought to the output connector so that any or all of the audio channels can be amplified by the on board amplifier or they can be fed by an external amplifier.

Because you have the three separate channels you could, if you felt particularly dextrous, wire the output up for stereo so, say, channel 1 came out of the left speaker, channel 2 came out of the right and channel 3 was mixed equally between the right and left speakers.

The unit also provides two 8 byte ports which can be controlled by the computer. They could, for instance, read an external keyboard or paddles. Each chip is fairly easy to program by PEEKS and POKES. Full instructions are provided.

The chip is available for £16.90, including p and p, from Bolton Electronics, 44 Newland Drive, Bolton BL5 1DP, Lancs. (Bolton 64772).

Get a load of this

Fulcrum Products, who produced the 'ZX81 Keyboard Bleeper' have developed a new product, the 'ZX Loading Aid'. It is designed so that you can set the cassette player volume control at just the right setting so that the signal the computer receives is neither too weak nor too strong. It is designed to ensure that even tapes made on 'foreign' cassette players will load first time.

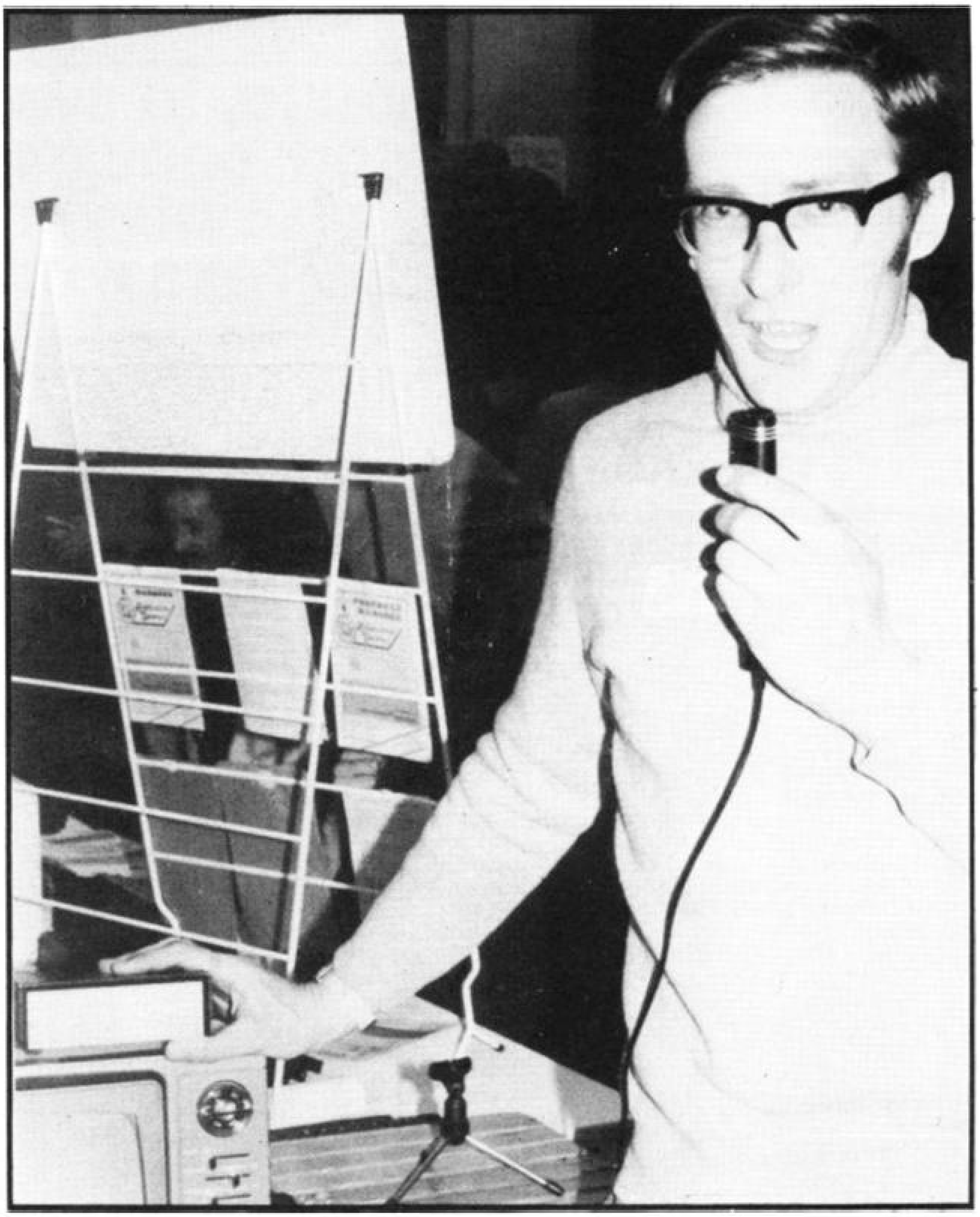
The Loading Aid is based on a circuit designed by Charles Rowbotham, which detects and shows the signal level on two LED's, enabling you to distinguish between quiet passages, voice introductions, the introductory buzz and the main body of the program. You can also actually see drop-outs in the tape.

The Loading Aid is a small black metal box which houses the circuit, two sockets of the size of those on the ZX81, and two jack plugs. As well as this, there is a red and a green Light Emitting Diode. You simply fit the Loading Aid between your cassette recorder plus power supply and the ZX81. When the tape is playing, you adjust the volume control so that the intensity of the green LED matches that of the red one. The cassette player is then at the optimum setting for that particular taped program. The ZX81 Loading Aid is £9.95, including p and p, and VAT, and is also available for the ZX80, or Spectrum. Specify which computer you have — Fulcrum Products, 'Hillside', Steep Lane, Findon, Worthing, West Sussex, BN14 0UF (090 671) 2750.

Fulcrum Products have also announced a new improved version of their Bleeper. When any of the normal or shifted keys is pressed, the unit gives a distinctive 'bleep'. That is, all 210 characters give a bleep. The unit is £8.95 which includes p & p and VAT.

The module is made up of a printed circuit board which fits into the ZX81, so there are no trailing wires outside the case. Both ZX81 keyboard tails simply plug into the module. Fulcrum provide two flexible ribbon cables to complete the connection back to the ZX81 pcb.

Features of this unit include the fact that no soldering is required; the model is small enough to fit under the keyboard; you can get an optional on/off switch for £1 extra; and the bleeper can also be used in conjunction with many of the full-size keyboards presently on the market for the ZX81.



William Stuart-Bruges talks to his attentive ZX81 through the 'Big Ears' speech recognition system

The Yellow ZX of Eighty

From Vaxjo, in Sweden, Lars Johansson, sent us this great program to play 'The Yellow Rose of Texas' on a ZX80.

As you can see, the music information is held within the string, A$, in line 50 which is checked element by element. Lines 160 and 310 strip the string down, character by character.

Note that there must be a semi-colon after the word PRINT in line 110. Lines 130 to 143 are just CLS.

Line 170 terminates, and lists the program after you've finished.

You can easily adapt this program to play other splendid melodies, by changing the contents of A$. We'd be very interested to see any other samples of music you can create.



PROGRAM LISTING

```
 10 LET A=17200
 20 POKE A,237
 30 POKE A,65
 40 POKE A,201
 50 LET A$='' R4R311114S46868CS8
    F88864S146664S6 411114S6868CS8
    F888644146346S8 R8R833333S33444 6S8
    F88864S146664S6 R4R311114446 868CS8
    RFRF88864S1R4R46346S8''
 60 LET B=CODE(A$)-28
 70 IF B=-28 THEN GOTO 200
 80 IF B>26 THEN GOTO 300
 90 LET E=2
100 FOR C=1 TO E* 34* (22-B)/(45-B)
110 IF USR(A)=1 THEN PRINT ;
120 GOTO 145 - B
130 CLS
131 CLS
132 CLS
133 CLS
134 CLS
135 CLS
136 CLS
137 CLS
138 CLS
139 CLS
140 CLS
141 CLS
142 CLS
143 CLS
150 NEXT C
160 LET A$=TL$(A$)
170 IF A$="" THEN LIST
180 GOTO 60
200 FOR D=1 TO 100
210 NEXT D
220 GOTO 160
```

```
300 LET E=(B-27)*3+1
310 LET A$= TL$(A$)
320 LET B=CODE(AS)-28
330 GOTO 100
```

This program will help you find your way through a long ZX80 program. To start it, you just type in RUN 9900. The ZX80 will then ask for the string it is to search for.

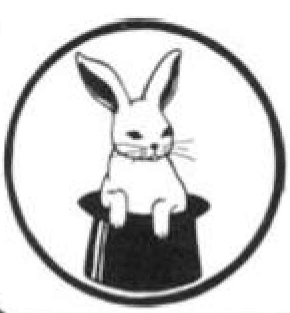
You type in the string to search (which can include shifted keys), or tokens (which you must prefix with a £ sign).

Here's an example. If you wanted to find 'POKE PEEK (A)...', you'd just answer '£PEEK(A)'. The program will then list the first appearance of the search string. To continue the search to find the next appearance of the string, enter GOTO G (that is, press the G key twice), and then NEWLINE. When you reach a NOT FOUND, GOTO G will start the next search at the top (first line).

You can stop the program by entering '$$'.

As a test, you might like to get the program to list all the TL$ in itself.

```
9900 DIM F(20)
9903 LET P=16426
9904 LET B=PEEK(16393)+256+PEEK(16392)
9905 PRINT "ENTER STRING TO SEARCH."
9907 INPUT F$
9908 IF F$="$$" THEN STOP
9909 IF F$'''' THEN GOTO 9920
9910 LET P=16426
9911 LET L=0
9912 IF F$="" THEN GOTO 9920
9913 LET L=L+1
9914 LET F(L)=CODE(F$)
9915 IF NOT F(L)=12 THEN GOTO 9918
9916 LET F$=TL$(F$)
9917 LET F(L)=CODE(F$)+192
9918 LET F$=TL$(F$)
9919 IF L<20 THEN GOTO 9912
9920 IF L=0 THEN GOTO 9907
9930 FOR X=1 TO L
9931 IF NOT PEEK (P+X-1)=F(X) THEN GOTO 9940
9932 NEXT X
9933 LET X=P
9934 LET X=X-1
9935 IF NOT PEEK(X)=118 AND X>16422 THEN GOTO
     9934
9936 LET X=PEEK(X+1)x256+PEEK(X+2)
9937 LET P=P+1
9938 LET G=9930
9939 LIST X
9940 LET P=P+1
9941 IF P<B THEN GOTO 9930
9942 CLS
9943 PRINT "(";
9944 FOR X=1 TO L
9945 PRINT CHR$(F(X));
9946 NEXT X
9947 PRINT ")NOT FOUND."
9948 PRINT
9949 GOTO 9903
```

Subscriptions

Make sure you get every issue of *ZX Computing* – Now bi-monthly!



Just £11.50 will ensure the next six issues will be lovingly wrapped and posted to you. Just fill in the form below, cut it out and send it with your cheque or postal order (made payable to ASP Ltd) to:-

ZX Computing Subscriptions
513 London Road,
Thornton Heath,
Surrey CR4 6AR

Alternatively you can pay by Access or Barclaycard in which case simply fill in your card number, sign the form and send it off. Do NOT send your card!

Make the most of your ZX computer with *ZX Computing* – Now bi-monthly!

Subscription Order Form

Cut out and SEND TO:

ZX COMPUTING Subscriptions
513, London Road,
Thornton Heath,
Surrey, England.

Please commence my subscription to ZX Computing with the very next issue.

SUBSCRIPTION RATES

(tick – as appropriate)

£11.50 for six issues UK
£13 for six issues overseas surface mail.
£23.80 for six issues overseas airmail.

I am enclosing my (delete as necessary) Cheque/Postal Order/International Money Order for £..........
(made payable to A.S.P. Ltd)
OR
*Debit my Access/Barclaycard**
*(*delete as necessary)*

Please use BLOCK CAPITALS and include post codes.

Name(Mr/Mrs/Miss) ....
delete accordingly

Address ....

....

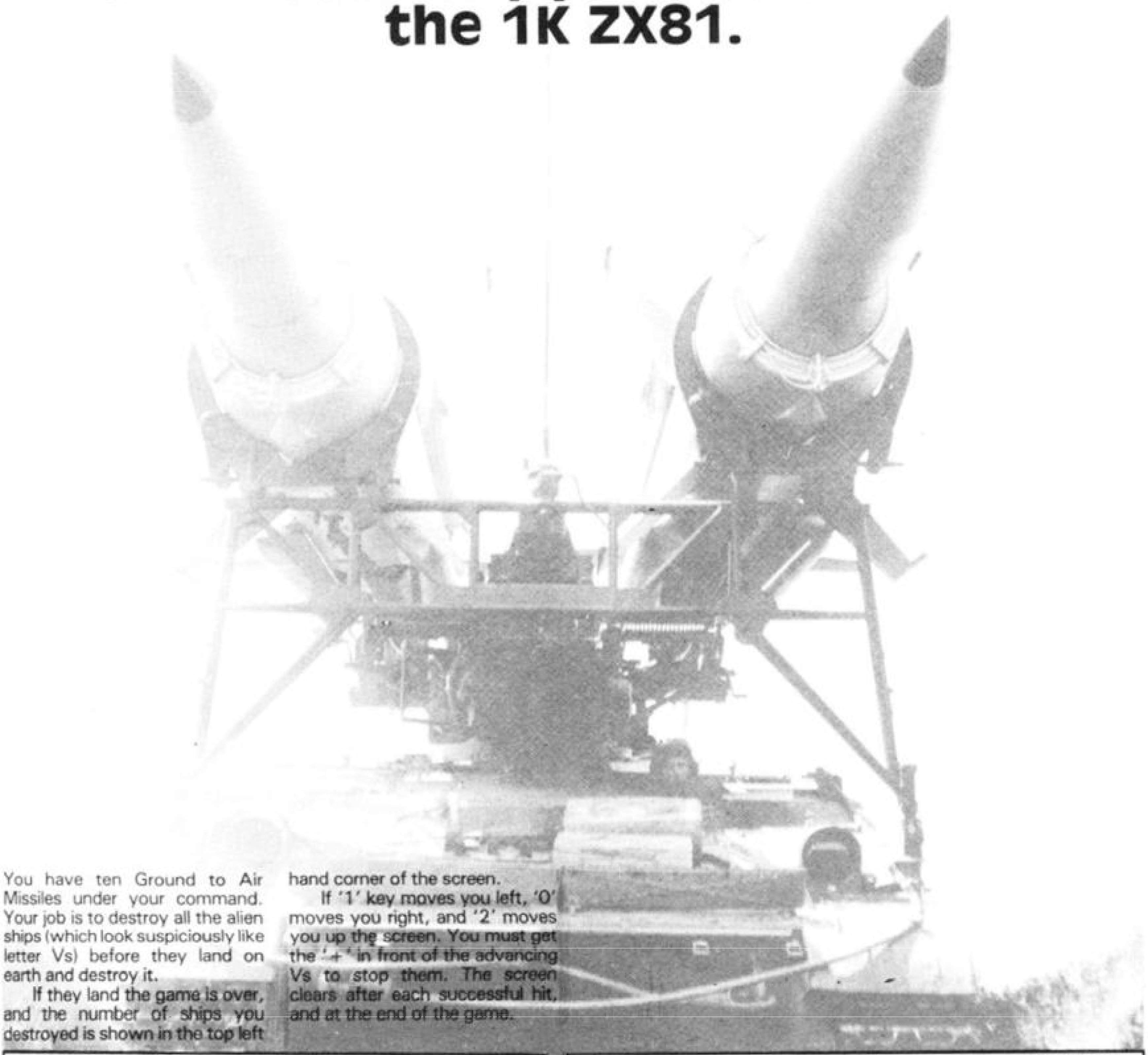
....

Signature ....

Date ....

Ground to Air Missile

From Cork in Ireland, Aidan Walsh and Kevin MacCarthy present G.A.M. for the 1K ZX81.



You have ten Ground to Air Missiles under your command. Your job is to destroy all the alien ships (which look suspiciously like letter Vs) before they land on earth and destroy it.

If they land the game is over, and the number of ships you destroyed is shown in the top left hand corner of the screen.

If '1' key moves you left, '0' moves you right, and '2' moves you up the screen. You must get the '+' in front of the advancing Vs to stop them. The screen clears after each successful hit, and at the end of the game.

```
  1 FOR N=PI-PI TO VAL "9"
  2 LET A=VAL "20"
  3 LET B=VAL "11"
  4 LET X=PI-PI
  5 LET Y=B
  6 FOR F=PI-PI TO VAL "63"
  7 PLOT F,PI-PI
  8 NEXT F
  9 PRINT AT A,B;CHR$ 21
 10 PRINT AT X,Y;CHR$ 59
 11 PRINT AT A,B;CHR$ 0
 12 LET Z=INT (RND*VAL "3")
 13 LET X=X+1
```

```
 14 LET Y=Y+(Z=VAL "1")-(Z=VAL "0")
 15 IF INKEY$="9" THEN LET A=A+2
 16 LET B=B+(INKEY$="0")-(INKEY$="1")
 17 LET A=A-(INKEY$="2")
 18 IF A=X AND B=Y OR X>=22 THEN GOTO VAL "20"
 19 GOTO VAL "9"
 20 CLS
 21 IF A=X THEN NEXT N
 22 IF X=VAL "22" THEN PRINT N
```

Moving with the flow

If you're bogged down with a bug, a flowchart can help. Henry Budgett, editor of 'Computing Today', tells you how to go about it.

People who program generally tend to fall into one of two categories, those who use flowcharts and those who don't. I tend to write mine after the program and then correct the bugs, and I'm sure many of you do too!

The techniques of flowcharting are of great benefit to those who like to tackle problems logically, they draw vast diagrams, test for all the possible quirks and then code up the results. The result of all this is usually a superb program, it never fails and is always late.

The rest of us write and debug our efforts as we key them in, end up with programs that work, fail occasionally and are usually ready on time. In this article I hope to put across some of the ideas behind the writing of flowcharts and demonstrate their useful points.

The Simple Idea

A flowchart is defined as "A diagrammatic representation of a series of events, usually in dicating the analysis or solutior of a problem[1]." This is simila to, but not quite the same as ar Algorithm, this is defined as "A defined process or set of rule for solving a given problem[1]."

One usually starts with ar algorithm, produces the flowchart and then codes the program. The simplest form of

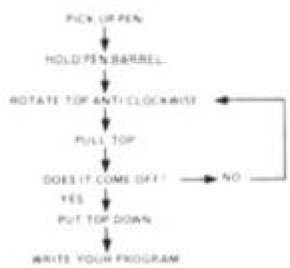


Fig 1. A Flowchart using no special symbols.

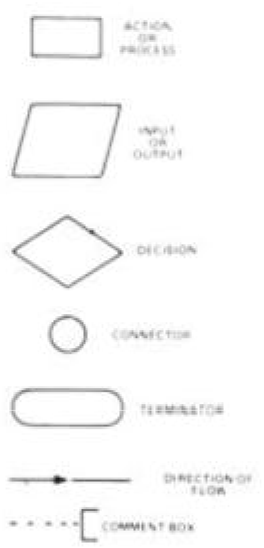


Fig 2. The standard flowcharting symbols.

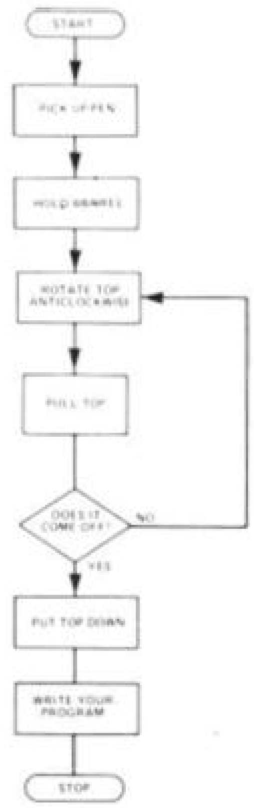


Fig 3. Figure 1 redrawn using the standard symbols.

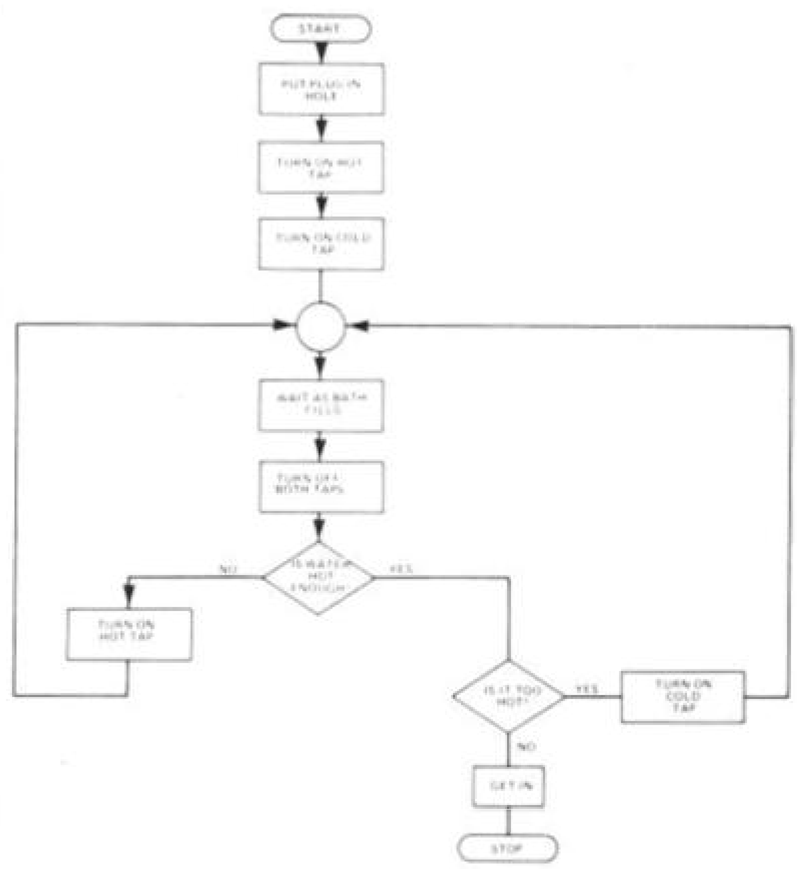


Fig 4. An attempt to flowchart a more complex problem.

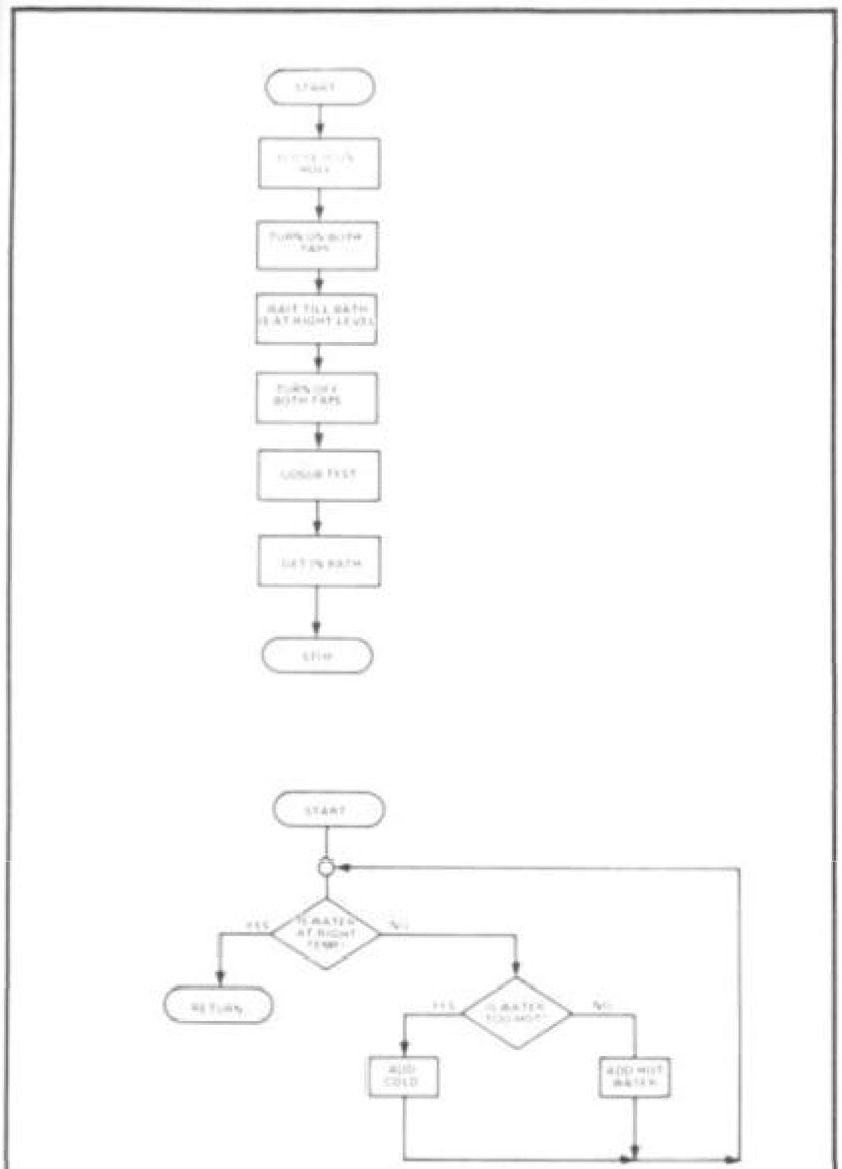


Fig 5. Splitting the problem can often make life easier.

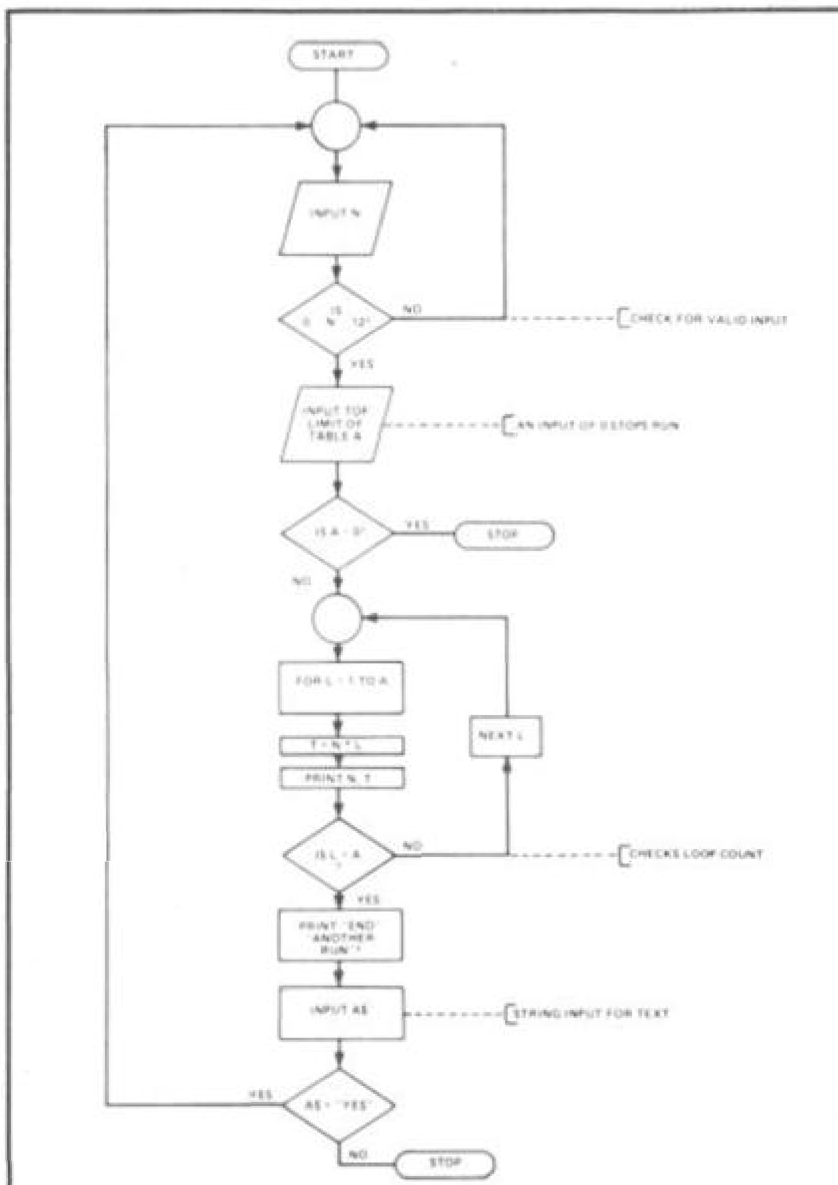


Fig 6. A true program flowchart for a simple task.

flowchart is shown in Fig. 1, it uses no special symbols, and is really an extended version of the basic algorithm.

Flowcharts usually contain lots of pretty little boxes which must mean something, and indeed they do. In Fig.2 I have listed all the common types and their designated functions. This is only a small selection of the available symbols but for most purposes it will be quite adequate.

The Standard Use

Having taken a look at the available set of symbols we can now re-write our simple flowchart in acceptable form, this is shown in Fig. 3. For the actual task of converting it into a given language this will be quite sufficient, regardless of which language is to be used.

A problem of this staggering complexity doesn't really deserve a flowchart at all, and indeed most proficient amateur programmers are quite capable of coding up large programs from a simple set of rules, or even the basic algorithm.

In Fig.4 I have attempted to flowchart another everyday problem, that of running a bath. As can be quickly seen it will work but is by no means bug-proof. Never mind, we'll sort them out later is the usual reply, in fact it's quite good enough to write a program from.

We will take a last look at this program flowchart before we move on — it can be re-written into two parts, a Control section and a single subroutine section of the task as subroutines with their own flowcharts. One can quickly sort out complex problems, and even write and test the various routines on their own before fitting them into the complete program.

The Real World

Computers being what they are, logical, the previous attempts at flowcharting bear no relation to a true programmers flowchart.

A typical example of such a beast can be seen in Fig. 6. The task is to produce a set of arithmetic tables for any given number between 1 and 12. The diagram shows all the steps needed and you should be able to follow it through on your own, there are comments!

The ideal of every programmer is to produce not only the ultimate bomb proof program but also to have it lavishly documented. This is the breakpoint between professional programs for a software house, or indeed a magazine for publication, and hopefully payment.

It is almost obligatory to include not only a flowchart but a complete description of just what it does. In a case such as this you will find that your first flowchart will be so scrawled on that you have to re-draw it and it is well worth investing in a stencil that gives the standard symbols.

It is also essential to keep a duplicate set of all the documentation for security, if you lodge a sealed set with the bank you have got a handy piece of evidence in case anyone rips off your version of Pacman and starts selling it and not paying any royalties!

In Conclusion

If you are capable of determining the way you wish to solve any given problem, writing the algorithm, you are capable of producing a flowchart.

They are useful for debugging programs but you will find that they soon become covered with modifications and have to be re-drawn.

Their most useful function is as a piece of documentation, how often do you remember how a program worked after six months, and as a means of testing out sections of a program such as subroutines.

Flowcharts are not essential as some people would have you believe but they do bridge the gap between successful programs and those which work.

References

[1] Both definitions are taken from The Dictionary of Data Processing from Newnes Butterworths so you can argue with them!

MICHAEL ORWIN'S ZX81 CASSETTES

THE BEST SOFTWARE (BY VARIOUS AUTHORS) AT LOW PRICES

QUOTES

"Michael Orwin's £5 Cassette Two is very good value. It contains 10 stolid well designed games which work, offer plenty of variety and choice, and are fun."
from the ZX Software review in *Your Computer*, May '82 issue

"I had your Invaders/React cassette . . . I was delighted with this first cassette."
P. Rubython, London NW10

"I have been intending to write to you for some days to say how much I enjoy the games on 'Cassette One' which you supplied me with earlier this month."
E. H., London SW4

". . . I previously bought your Cassette One and consider it to be good value for money!"
Richard Ross-Langley
Managing Director
Mine of Information Ltd.

CASSETTE 1
(eleven 1k programs)

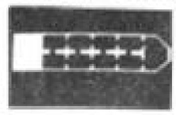
machine code:
React, Invaders, Phantom aliens, Maze of death, Planet lander, Bouncing letters, Bug splat.
Basic:
I Ching, Mastermind, Robots, Basic Hangman PLUS Large screen versions of Invaders and Maze of Death, ready for when you get 16k.
Cassette One costs £3.80

CASSETTE 2
Ten games in Basic for 16k ZX81

Cassette Two contains Reversi, Awari, Laser Bases, Word Mastermind, Rectangles, Crash, Roulette, Pontoon, Penny Shoot and Gun Command.
Cassette Two costs £5

CASSETTE 3
8 programs for 16k ZX81

STARSHIP TROJAN



Repair your Starship before disaster strikes. Hazards include asphyxiation, radiation, escaped biological specimens and plunging into a Supernova.

STARTREK This version of the well known space adventure game features variable Klingon mobility, and graphic photon torpedo tracking.

PRINCESS OF KRAAL An adventure game.

BATTLE Strategy game for 1 to 4 players.

KALABRIASZ World's silliest card game, full of pointless complicated rules.

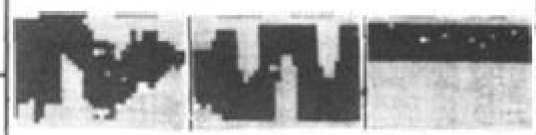
CUBE Rubik Cube simulator, with lots of functions including 'Backstep'.

SECRET MESSAGES This message coding program is very txlp qexi jf.

MARTIAN CRICKET A simple but addictive game (totally unlike English cricket) in machine code. The speed is variable, and its top speed is very fast.
Cassette 3 costs £5

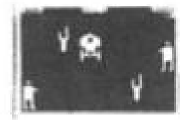
CASSETTE 4
8 games for 16k

ZX-SCRAMBLE (machine code)

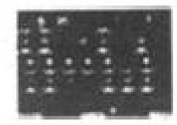


Bomb and shoot your way through the fortified caves.

GUNFIGHT (machine code)



INVADERS (machine code)



GALAXY INVADERS (machine code)
Fleets of swooping and diving alien craft.

SNAKEBITE (machine code)
Eat the snake before it eats you. Variable speed (very fast at top speed).

LIFE (machine code)
A ZX81 version of the well known game.

3D TIC-TAC-TOE (Basic)
Played on a 4x4x4 board, this is a game for the brain. It is very hard to beat the computer at it.

7 of the 8 games are in machine code, because this is much faster than Basic. (Some of these games were previously available from J. Steadman).
Cassette 4 costs £5

FUNGALOIDS (Machine code)

Recorded on quality cassettes, sent by first class post, from:
Michael Orwin, 26 Brownlow Road, Willesden, London NW10 9QL (mail order only please)

NEW softwares for **NEW** computer

ZX SPECTRUM ZX81

SOUND & COLOUR ON ALL ZX SPECTRUM PROGRAMS

SEA WAR: (ZX-sp **£6.00** ZX81 **£5.00**)
Completely new designed game for one or two players. Attractive screen display and super control.

IQ GAME PACK I: (ZX-sp **£4.00**; ZX81 **£3.50**)

ADDER GAME = (ZX-sp **£5.00**; ZX81 **£4.50**)
A new idea in which maths is involved in this fast action game.

Send s.a.e. for full details. Mail order only.
All cheques and postal orders made payable to:

Panda Software,
51 Elgin Street, Shelton, Stoke-on-Trent ST4 2RD

Please supply

Game	ZX-sp	ZX-81
Sea War ______	at **£7.00** ______	at **£6.00**
IQ Game Pack I ______	at **£4.00** ______	**£3.50**
Adder ______	at **£5.00** ______	at **£4.50**

I enclose **£** ______ for above items.
My address is ______
______
My name ______ Signed ______

All prices include VAT and P&P. U.K. delivery: Allow up to 28 days.

SOUND with ZX-81!

MAKE AMAZING SOUND EFFECTS WITH YOUR ZX-81





£25.95 incl p&p & VAT **THE ZON X-81**

- The ZON X-81 SOUND UNIT is completely self-contained and especially designed for use with the ZX-81. It just plugs in – no dismantling or soldering.
- No power pack, batteries, leads or other extras.
- Manual Volume Control on panel – ample volume from built-in loudspeaker.
- Standard ZX-81 – 16K Rampack or printer can be plugged into ZON X-81 Sound Unit without affecting normal ZX-81 operation.
- Huge range of possible sounds for games or: Music, Helicopters, Sci-Fi, Space Invaders, Explosions, Gun-shots, Drums, Planes, Lasers, Organs, Bells, Tunes, Chords etc., or whatever you devise!
- Uses 3-channel sound chip giving programme control of pitch, volume of tones and noise, all with envelope control.
- Easily added to existing games or programmes using a few simple 'BASIC' lines.

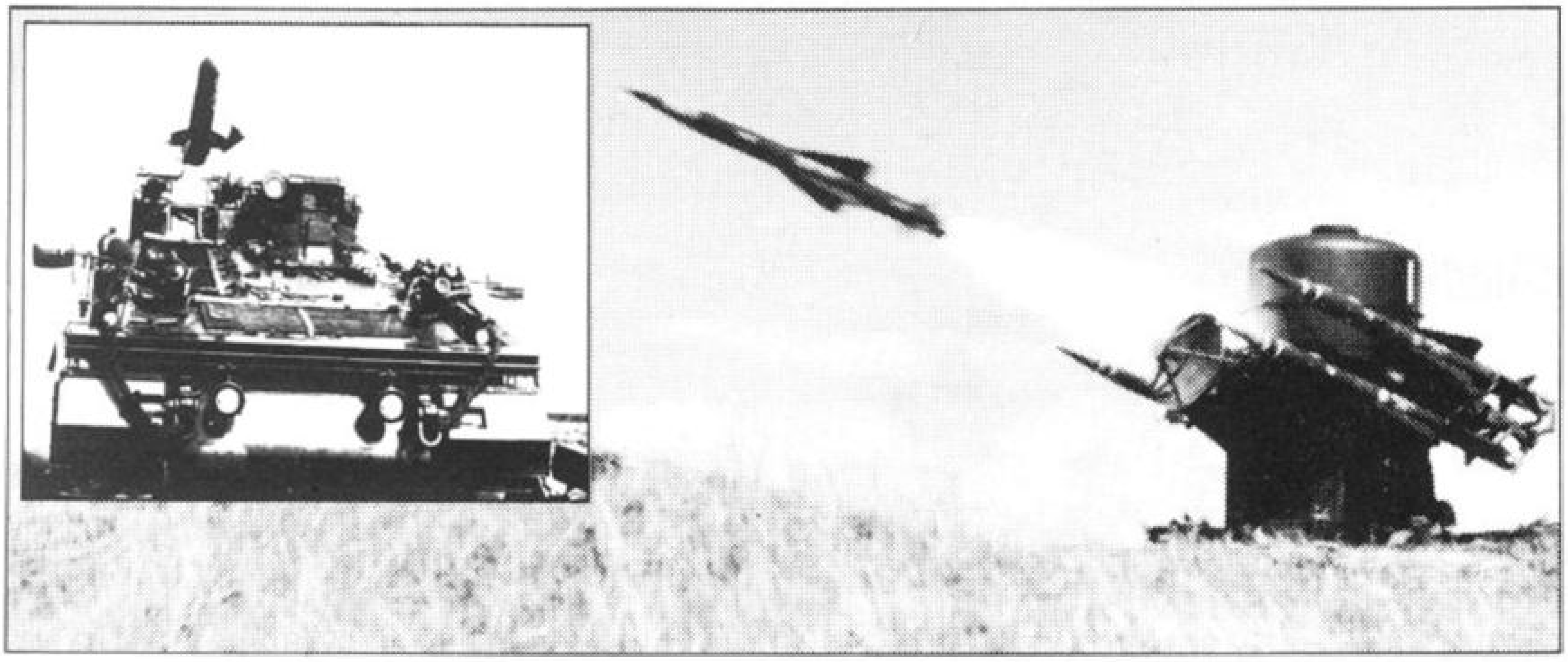
FULL instructions with many examples of how to obtain effects and the programmes, supplied. Fully Guaranteed.

BI-PAK Dept SU, P.O. Box 6, 63A, High Street, Ware, Herts.

Access & VISA accepted. Ring 0920 3182 for immediate despatch.

And what is your defence

Defending the earth in three dimensions sounds pretty impressive. Thirteen-year-old Joseph Nicholson from Chilton tried out the latest offering from J K Greye Software.



The program loaded with no trouble, and after about four minutes the program auto-ran. Impressive instructions rolled up the screen telling me that I was the only space ship that the planet had and I must defend it to the last. Who, me? They must be joking!

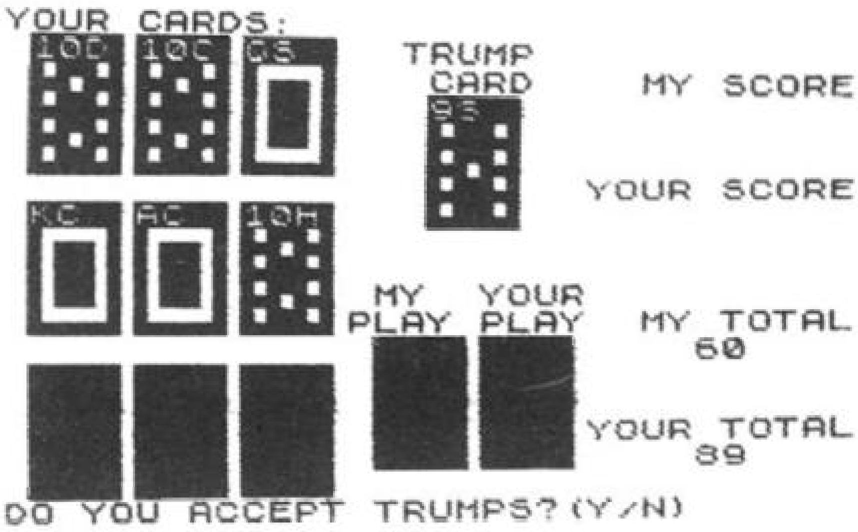
When I bought this game I was under the impression that this was just an upmarket version of the usual defender games, the idea of 3D Defender hadn't really clicked in my mind. The screen in 3D Defender is what you would see if you were looking through the cockpit of the space craft. The aliens actually fly towards you in full 3D, getting larger as they get nearer. A few instruments appear on the screen as well: a radar, an altimeter, a proximity meter (how near you are to an alien space craft), the number of shields you have and your score.

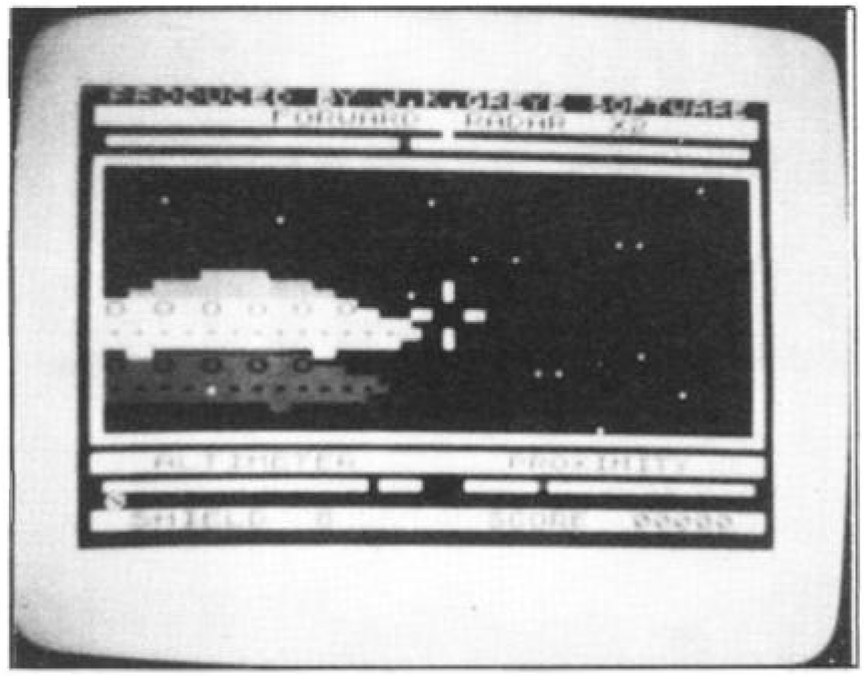
After pressing newline the game began. The graphics was outstanding. After getting myself accustomed to the non-standard movement keys (the game does not use the cursor keys to move, but the movement keys are arranged to stimulate joystick control. Once this has been mastered, the game feels much more realistic) I decided to try and "save the planet from the marauding alien space craft" as the instructions for the cassette put it. This was nowhere near as easy as it seemed. Whenever I got near to the beggers they would either shoot their Plasmo at me and shoot me down or I would actually collide with one of their space craft instead of shooting it. Every now and again meteors would zoom across the screen and if you hit one of them one of your lives would be lost. After about one hour, yes, one hour!, I shot my first alien. 250 the score read, but that didn't stay there for long as everytime an alien lands on the earth 50 points is deducted from the score. My score was reduced to 0 in about 30 seconds! It took me almost another hour to really get the hang of it, this is certainly not an easy game. In the instructions for the cassette they told you how to alter the speed of the game, you and the aliens. I decided to change the speed of the aliens (slower of course!). You get out of the program by pressing the EDIT key (the break key has no effect). This proved rather difficult, as the program recognised the SHIFT key as a key in its own right. I soon found that by pressing the 1 key (the key with EDIT written on it) without the shift depressed it worked perfectly. Inspection of the program revealed that the game was written almost completely in machine code with only 2 lines of BASIC. A SAVE line (which makes it RUN automatically upon loading) and a RAND USR line. Having POKEed the alien to its slowest speed, the game was still fast, but not so fast that I couldn't play properly.

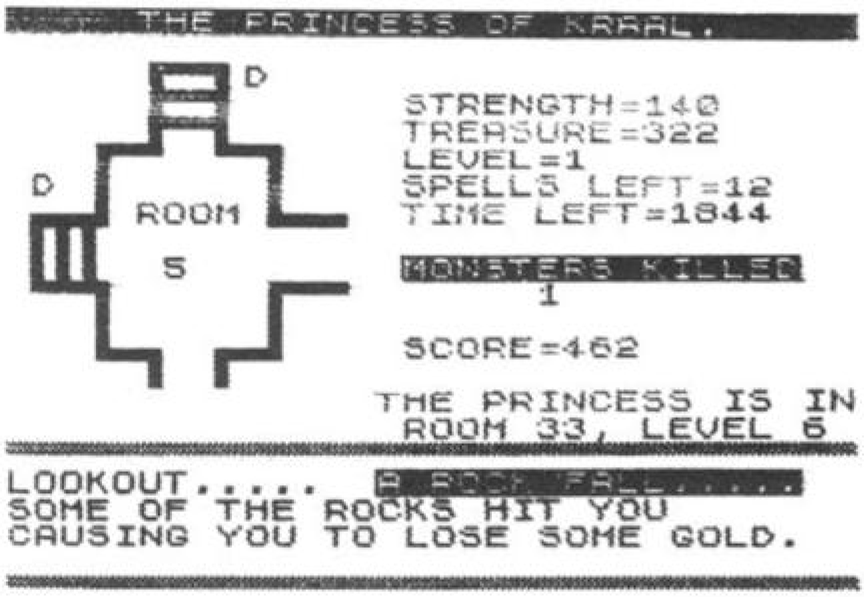
About the best way I can sum up this game is "GREAT"! The game is good value at £4.95, with graphics second only to the real arcade game. My only minus for the game is that it does take quite a long time to learn to play properly. But then, most games that are too simple become boring after a time anyway. Recommended.

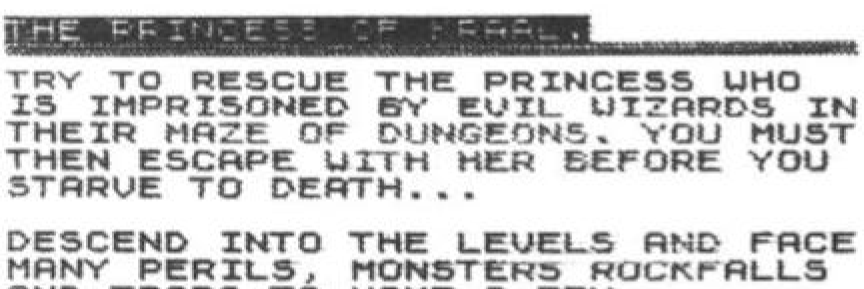
You have been seen gulping...

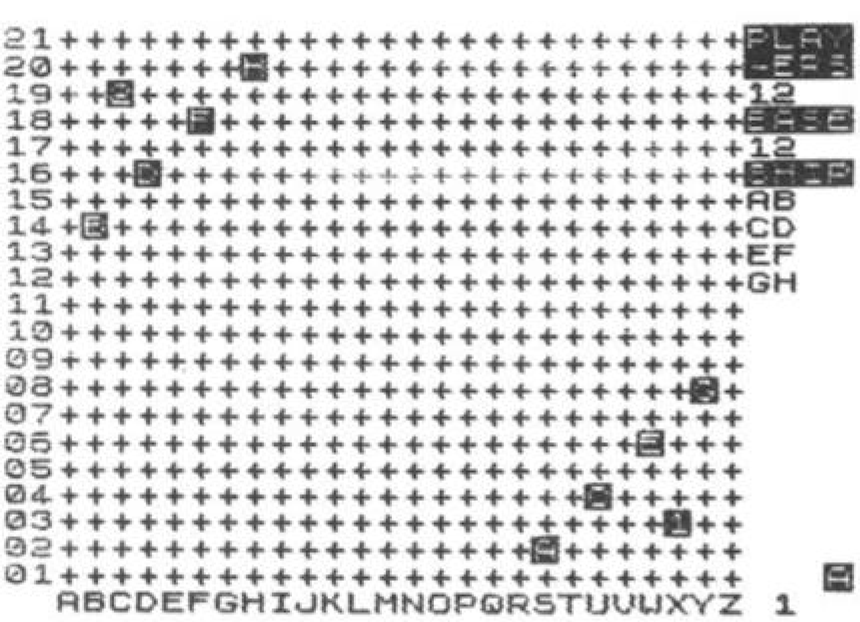
I bought Campbell Systems' GULP from W H Smith for £3.95. When I bought the game I was under the impression that this was a kind of one 'ghost' Pac-man game under a new name. The only 'instructions' on the cassette for playing the game was "Can YOU outwit the GRUESOME GULPER that seeks out... gives chase... accelerates... and devours!". I had not much to go on. The game seemed to be saved under a name other than "GULP" as when I typed LOAD "GULP", it didn't load, and when I typed LOAD "" it loaded very easily. It took about two minutes to load. The game auto-ran. A menu appeared, it said: "A...PLAY, B...MAZE, C...SPEED, D...GRADE, E...RESET, F...SAVE, PRESS G FOR INSTRUCTIONS." I pressed G. I was reassured that the game WAS a one 'ghost' version of Pac-man, the 'ghost' being called a 'chaser'. You have 5 lives and you have to eat up all the dots in the maze. The more you eat the faster he gets. You have a choice of 5 different mazes, wow! The cursor keys are used to move. You are an 'O' and you start in the middle of the screen (on all games apart from game 3 where you start at the left-hand side). The chaser is an inverse 'X' and he starts in the bottom right-hand corner. High scores are kept. Pressing newline again explained what the menu meant. The











speed of the game could be altered by pressing 'c' and the appropriate number (1-9), and his acceleration could be altered by pressing 'd'. I set the speed to 1 and the acceleration to 1 for the slowest game. The maze was already set at 1 so I pressed 'A' to RUN. The maze flashed up instantly, obviously written in machine code. It looked suitably complicated. The game moved at a reasonable pace. One point I noticed was that your movement stopped directly a finger was lifted from the key. This is different from the movement in the arcades and makes the game much harder.

The maze lacked things like power pills and tunnels, but the fact that you could choose one of 5 mazes and choose the speed and acceleration seemed to make the game just as good. In maze 5 the walls of the maze make the world 'GULP'! The game is very addictive and great fun, but I do have a few grumbles:

When the screen is cleared of dots the game does not display a new screenful. Instead it relies on you pressing the 'O' key to end the game.

Some mazes have more dots than others, so if you win maze 2 for instance, you could still beat that high score on maze 1. However, this problem is compensated for by a function that allows you to reset the high score.

My last grumble is that when a life is lost the chaser places another dot in the square that he started off from. This means that to obtain the best score you have to lose all your lives. This seems pretty idiotic.

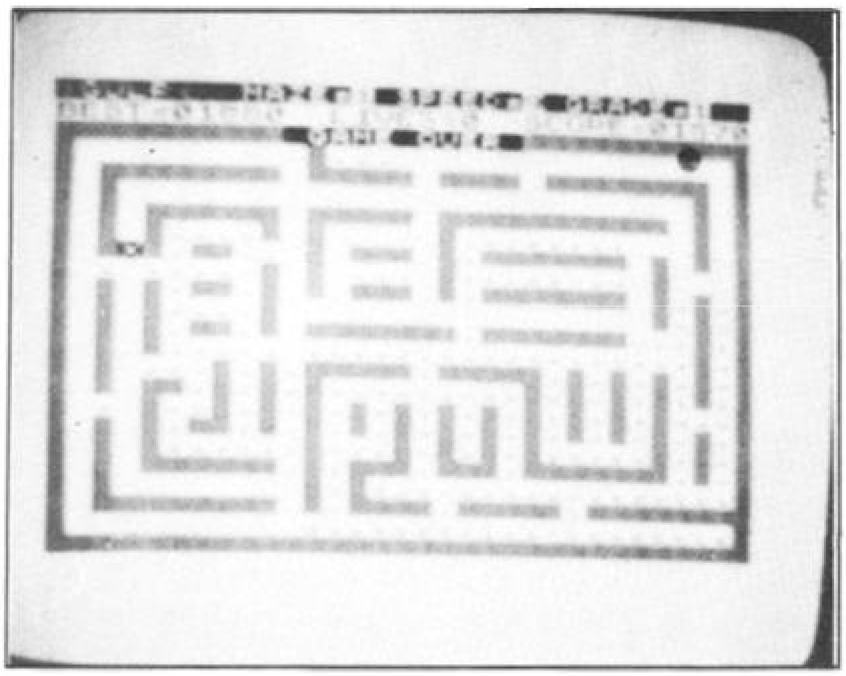
However, after all those grumbles I still stick to my statement that the game is exciting, addictive and great fun to play.

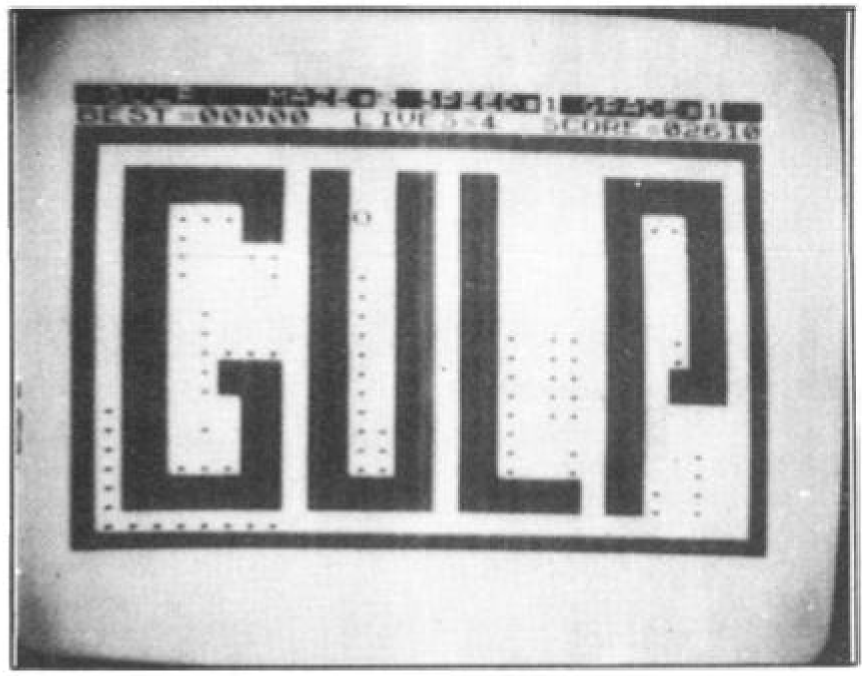
Michael Orwin wins again

From Loughton, 15-year-old James Walsh puts Orwin's Cassette Three through its electronic paces... and likes what he sees.

Eight programs on one cassette? Sounds ominous, last time I got one of these, the programs were so poor that they could have been copied out of the manual. Well, could this be different?

The first program is called "STARSHIP TROJAN" and the idea is that your starship is damaged and you have to repair it before becoming asphyxiated, dying of radiation, plunging into a supernova or some other gruesome death. The graphics on this are quite good, but the actual program was a bit slow for me. The second game is called 'STAR TREK' (original !!!). This is a good version of the well known (you said it) space game. Again because it was in BASIC I found it a bit slow but if you do not mind waiting around a little you may find it very good. The next program is called "PRINCESS OF KRAAL". Quite predictably this is another adventure game — well it is not just another! With a visual map, different levels and a whole host of dangers, let alone 100 different difficulty levels, it makes for a very addictive game. The idea of 'MARTIAN CRICKET' is to get from one side of the screen to the other without hitting the balls being constantly hurled at you.





Though this game is simple it also becomes addictive. Of the remaining four games the one that caught my eye first was "KALABRIASZ" which is quoted as being the worlds silliest card game, and it lives up to it's name! Next comes CUBE which is a very well written simulation program of that dreaded mind-bender.

Coming a very close seventh an eighth are BATTLE and SECRET MESSAGE, both of which I found reasonably good.

Conclusion

I cannot comment on the documentation as I did not receive any. But I am assured that it is about six A4 pages long and comprehensive (it would have to be for some of these games). Altogether it is a well produced, good value for money cassette with eight very good to mediocre games on it.

Cassette 4

Michael Orwin seems to have got into the habit of selling cassettes of eight games all of which could have been sold separately for six pounds a piece, whilst he sells all eight on one high quality cassette for £5. The only way I can do true justice to these programs is by looking at one on it's own as if it was a different cassette.

(i) ZX- Scramble; has been written and marketed by other people, but this is by far the best version I have seen for the ZX81. The graphics are excellent and the speed is incredible even for a machine-code program.

(ii) Gunfight. This is one of the arcade type games which I definitely have not seen before on the ZX81. Although the name is misleading the idea is that you are one of two cowboys on the screen and whilst not hitting the stage coach which moves steadily up the screen, or one of the many cacti, you must try to shoot your opponent. The graphics are excellent, and this is definitely the best one-or-two player graphics games that I have seen for the ZX81 to-date.

(iii) INVADER; Yes, I know, another invaders game and you've seen them all before, haven't you !! But this one is actually is better than any other I have seen on the ZX81. The graphics are far better as you have three characters for each invader. The only thing that I found difficult was that the game did not stop and restart when you are hit.

(iv) GALAXY INVADERS; this is a very good machine code version of the Galaxians game, with very 'pretty' sweeping aliens.

(v) SNAKEBITE; This is not quite as graphically mind blowing as the last four, but with the ability to build walls, lay anthills and vary the speed it becomes a very addictive game. Oh I nearly forgot, the idea of the game is to eat the snake's tail first before it eats you.

(vi) LIFE; A cleverly written version of the well-known game with good graphics.

(vii) 3D TIC-TAC-TOE. This is the only BASIC game on the tape, which means it is slower. But it is an advanced and addictive version of the game.

(viii) FUNGALOIDS. Last but definitely not least we come to the most original game, Fungaloids, which is easily as addictive as Invaders. Although the game is original there was no need to make the game so obscure! It has vague similarities to missile command, as the idea is to bomb the fungus as it grows, shedding spores. A weird but definitely wonderful game.

Conclusion

If each game was on a separate tape and selling for £5 each I would still recommend them. But all on one for £5.....! This sort of value for money just has not been seen before for any personal computer. It is interesting to note that out of the many software companies in this country, Michael Orwin is one of the few which has managed to continue to grow even after the Spectrum was announced.

Without sounding pushy I would like to conclude this review by saying — if you have a ZX81 and like games, then you should buy Michael Orwin's cassette 4.

Available by mail order only from: MICHAEL ORWIN, 26 Brownlow Road, Willesden, LONDON. NW10 9QL.

More pieces of eight

Following the recent launch of his first ZX81 cassette, Bargain Bytes One, Richard Shepherd is now offering a second; Bargain Bytes Two. As with the first cassette, it features eight varied progams for £5, but there are two extras. Firstly, a short test program at the start enables correct volume levels to be determined, and despatch is now promised within 24 hours of the order being received.

The new cassette features Seafaring Adventure in which the player commands a ship and tries to win promotion by his performance in battle. Skillful utilization of resources is vital. Supplies, men and ammunition must be carefully calculated and finely balanced.

When supplies run low, it becomes necessary to return to Port for them to be replenished. Naturally the journey is hazardous and the battles fierce, but a determined player can battle his or her way up the ranks to become First Sea Lord.

Other major games on the cassette are Stock Market, in which the player must make instant buying decisions as market information flashes appear, and Noughts and Crosses, which is on three levels; easy, beatable and impossible. There is also a 'Pub Style' Fruit Machine complete with random holds, systematic nudges and a winnings counter. A moving graphics Ski Run rounds up the games section.

On the educational front there is a General Knowledge Quiz, (with three levels) Copycat; an alphabetic Simon game, and a profit/break even point calculating money model.

Bargain Bytes Cassette Two is available now from Richard Shepherd, 22 Green Leys, Maidenhead, Berkshire, SL6 7EZ. Telephone (0628) 21107. Price £5 including postage and packing. All programs require 16K Ram Pack.

Michael Orwin's Cassette Three: Battle

J K Greye's '3-D Defender'

Campbell Systems' GULP:

Programming your computer for board games

There is one common thread which can hold together computer programs for such games as draughts, chess, reversi and even Nine Mens Morris. Tim Hartnell reveals the secret, and shows how it can be used to write an intelligent board game — from scratch.

Look first at diagram one. It shows a draughts or chess board, numbered to make it easy for a computer to handle. You can indicate any square on the board by referring to the number along the left hand side (such as 3), then the number along the top (such as 4). In this case, the lines numbered 3 (along the left hand side) and in the line numbered 4 (along the top) meet at the square numbered 34. If you wish to move a piece, you can do so by entering the number of the square you're moving from (such as 55), then the number you are moving to (such as 66), and the computer can understand exactly what you are doing. There is no need to change the numbers entered by the human player into another set of numbers in order that the computer can interpret them.

That's the first 'secret'. The second is that the board numbered in this way has another great advantage over a board which is simply numbered from one to 64 in order. When you move in any direction, no matter where you are on the board, the difference between the squares is the same.

I'll explain that somewhat cryptic statement. If you move one square up and to the right — like the move of a piece in draughts — you will move from, say, 24 to 35; or from 53 to 64; or from 71 to 82. But notice that no matter where you are on the board, the difference between your starting and ending squares is always

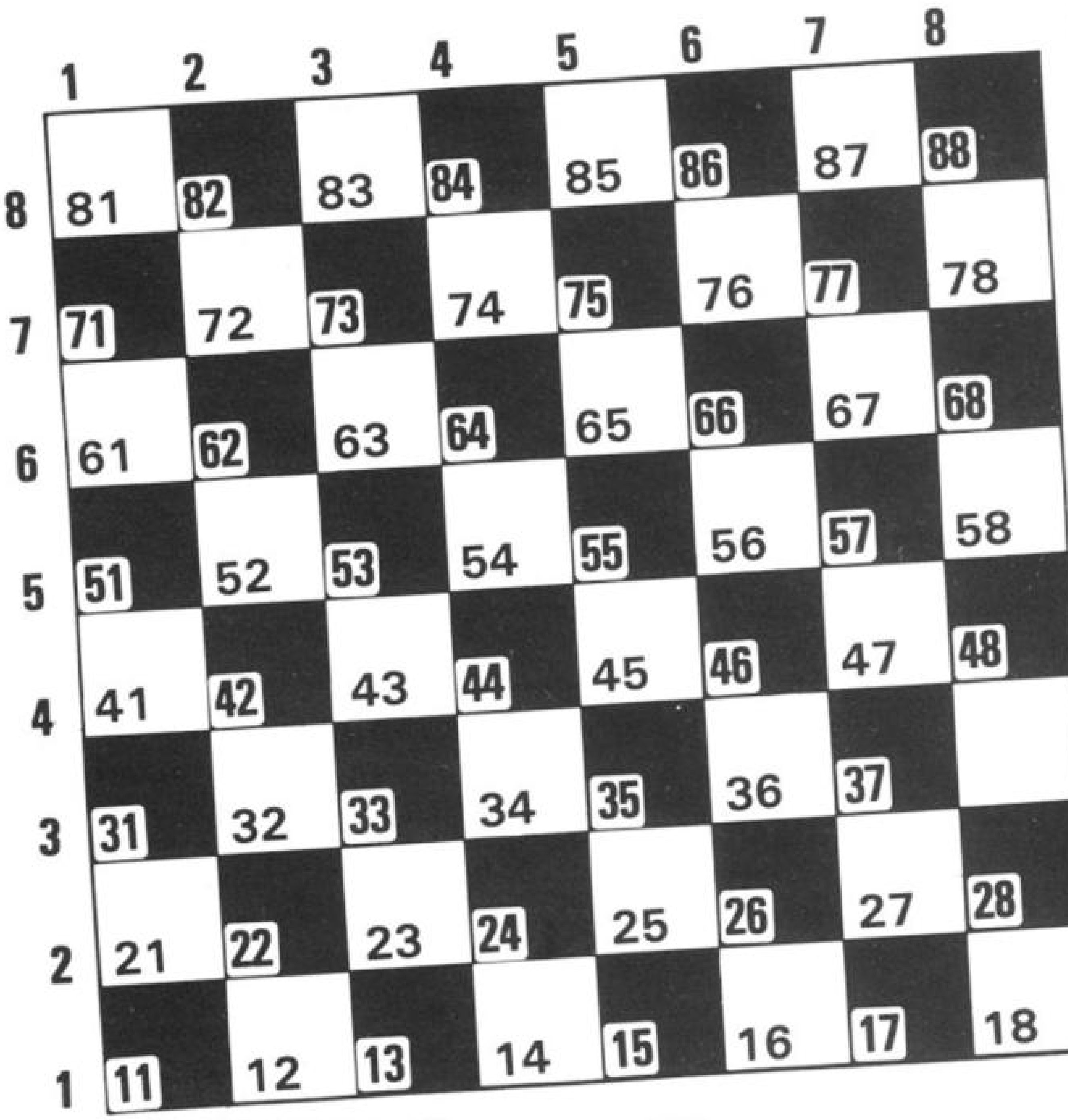


Fig 1 A chessboard game which a computer can use! All the squares are numbered up in ranks and files, so that the machine can be told exactly where to go!

11. If you move diagonally up to the left, you'll move from, say, 26 to 35 (plus 9), or 66 to75 (plus 9) or 22 to 31 (plus 9).

This predictability makes it relatively simple to create a board which the computer can handle.

Imagine the computer has a draughts piece on the square numbered 24. It could be programmed to check each square on the board, and every time it found one of its own pieces, could check if there was a human piece on the square numbered that (ie. 24 in our example) plus 11 (ie. on 35); and it could check to see whether the square 11 beyond that (ie. 46), was blank. If it found all these conditions were true, the computer could jump over square 35 into square 46, and capture the piece on 35.

This, in essence, is how many computer board games — from draughts, through Reversi to chess — work, based on a simple 8 x 8 grid numbered in this way.

If you were writing chess on this board, you could specify the moves of, for example, a knight, by knowing that it can always move to squares which are the following 'distance' from its own squares: 21, 12, −8, −19, −21, −12, 19 or 8. Try it now, by placing a coin on square number 55, and move it as if it was a knight, working out the mathematical relationship between the starting square, and the square you're moving to. You should find the differences are the same as the numbers just listed.

The Pieces

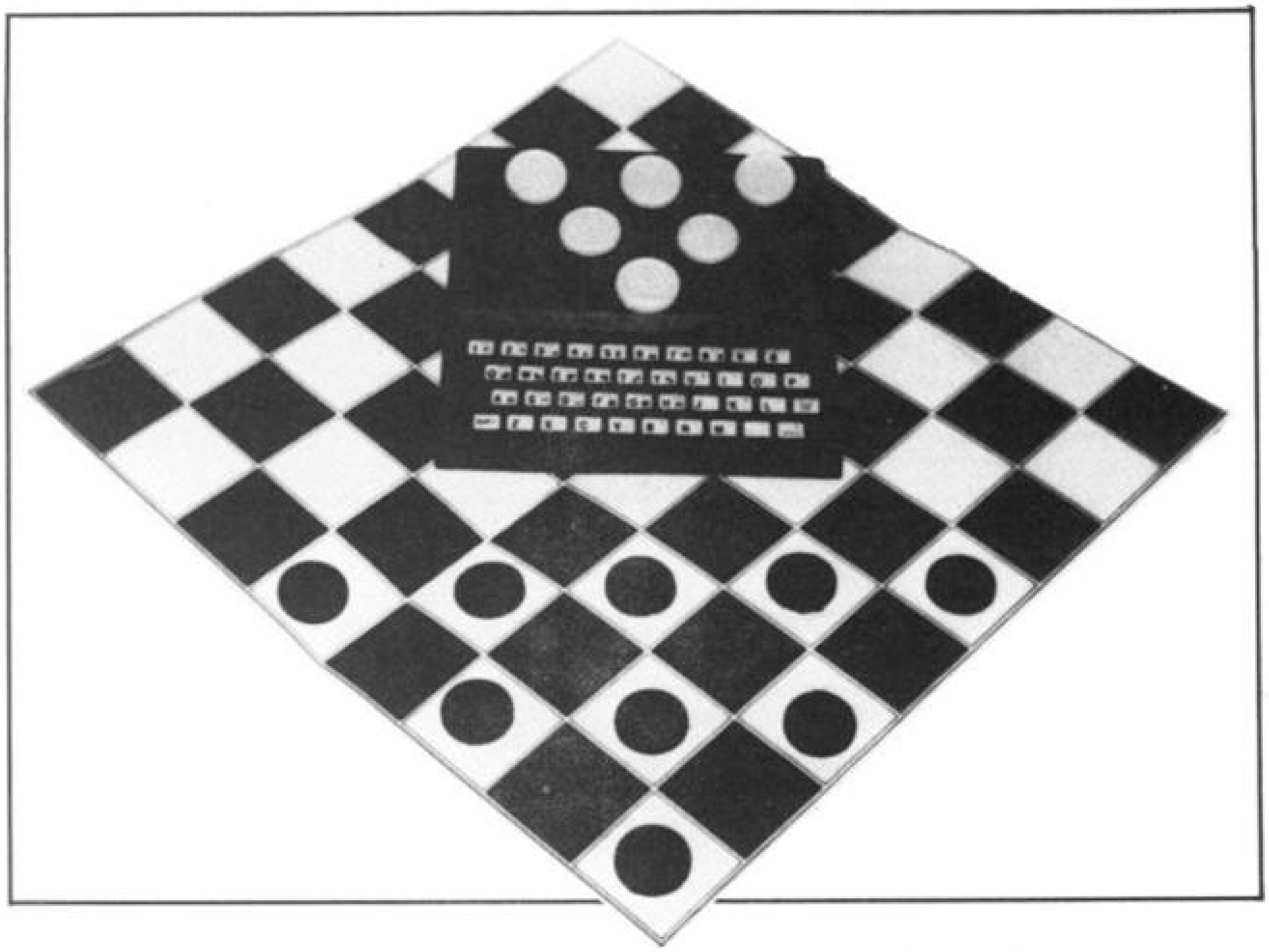
Let's move on now to produce a board game, making use of the information we've discussed so far in this article. We are going to write CORNER CHECKERS, which will be a game much the same as draughts, except that it is played by starting in the corners of the board rather than the ends, there are no multiple jumps, and no kings. Any piece may move in any diagonal direction. Captures are as in draughts, by jumping over an opponent's piece into an empty square, always moving on the diagonal. First we need an array to hold the pieces. We'll start the program with a title, and a GOSUB to send action to line 9000.

It is a good idea to assign the variables at the end of the program, as it makes the program run a little more quickly once the subroutine has been run (as it saves going right through the 'variables assignment' section every time the computer is going through the program, line by line, to find a GOTO or GOSUB address), and if you suddenly discover, when you're writing a program, that more variables are needed, there will be no shortage of places to put them, which there could well be if the variables were assigned at the start of the program.

The first 'mini-program' we'll enter, then, is program one. Next, we have to decide which squares on our board will be occupied by pieces, and what codes we will assign to those pieces. We'll be playing on the black squares in this game, starting the human pieces on 11, 13, 15, 22, 24, 31, 33, 42 and 51. The computer's pieces will be on squares 88, 86, 77, 68, 84, 75, 66, 57, 48. All other squares will be blank, and there will be — of course — other squares (such as those with numbers below 11 and above 88) which are off the board.

We need to assign the values to the elements of the A array, which we do by running through a loop, from one to 100.

Look at lines 9010, 9020, 9030 and 9040 in program two. These are acting as 'data statements', holding the



```
  10 REM CORNER CHECKERS
  20 GOSUB 9000
8990 STOP
9000 DIM A(100)
```

Program One: the opening lines of a ZX board game! The array is to hold the pieces in the memory of the ZX Computer.

```
9000 DIM A(100)
9010 LET H$="111315222431334251"
9020 LET C$="888677688475665748"
9030 LET B$="1214161821232527323
43638414345475254565861636567727
4767881838587"
9040 LET E$="8273645546372817263
544536271"
9050 FOR Z=1 TO 100
9060 LET A(Z)=9
9070 NEXT Z
9080 LET H=CODE "H"
9090 LET C=CODE "C"
9095 LET B=CODE "■"
9100 LET E=CODE " "
9105 FOR Z=1 TO 9
9110 LET A(VAL H$( TO 2))=H
9120 LET A(VAL C$( TO 2))=C
9130 LET H$=H$(3 TO )
9135 LET C$=C$(3 TO )
9140 NEXT Z
9150 FOR Z=1 TO 32
9160 LET A(VAL B$( TO 2))=B
9170 LET B$=B$(3 TO )
9180 NEXT Z
9190 FOR Z=1 TO 14
9200 LET A(VAL E$( TO 2))=E
9210 LET E$=E$(3 TO )
9220 NEXT Z
9230 LET COMP=0
9240 LET HUM=0
9250 PRINT AT 5,0;"DO YOU WANT T
HE FIRST","MOVE (Y OR N)?"
9260 INPUT U$
9265 CLS
9270 IF CODE U$<>CODE "Y" THEN G
OTO 50
9500 RETURN
```

Program Two: this section of the program holds the starting position of all of the pieces in the game. It also assigns names to the main variables used.

```
8000 REM **PRINT BOARD**
8010 PRINT AT 3,0;"COMPUTER > ";
COMP;TAB 16;HUM;" < HUMAN"
8020 PRINT AT 5,6;"▒▒▒▒▒▒▒▒▒▒▒▒"
8030 PRINT TAB 6;"▒▒12345678▒"
8040 FOR Z=8 TO 1 STEP -1
8050 PRINT TAB 6;"▒";Z;"▒";
8060 FOR X=1 TO 8
8070 PRINT CHR$ A(10*Z+X);
8080 NEXT X
8090 PRINT "▒"
8100 NEXT Z
8110 PRINT TAB 6;"▒▒12345678▒"
8120 PRINT TAB 6;"▒▒▒▒▒▒▒▒▒▒▒"
8125 IF COMP=7 THEN PRINT AT 0,0
;"I WIN";END
8126 IF HUM=7 THEN PRINT AT 0,0;
"YOU WIN";END
8130 RETURN
8990 STOP
```

Program Three: a central part of our checkers game! This little routine prints out the board and the position of all of the pieces. Don't forget to add on line 30 before you try to run this section!



```
7000 REM **PLAYER MOVE**
7010 PRINT AT 19,0;"ENTER YOUR M
OVE AS ""3344"""
7020 INPUT A$
7030 IF LEN A$<>4 THEN GOTO 7020
7040 PRINT AT 19,0;"
"
7050 LET A=VAL A$(1 TO 2)
7060 LET B=VAL A$(3 TO 4)
7070 LET A(B)=A(A)
7080 LET A(A)=E
7090 IF ABS (A-B)>11 THEN LET A(
(A+B)/2)=E
7100 IF ABS (A-B)>11 THEN LET HU
M=HUM+1
7900 RETURN
```

Program Four: the easiest of all the subroutines employed in the game is that to make and record your move, ie. the 'human' move. You must add line 40 to Program One in order to utilise the routine in the game.

```
6000 REM **COMPUTER MOVE**
6010 FOR Z=88 TO 11 STEP -1
6020 IF A(Z)=C THEN GOTO 6050
6030 NEXT Z
6040 GOTO 6200
6050 LET Y=-11
6055 IF Z+Y>88 OR Z+Y<11 OR Z+2*
Y>88 OR Z+2*Y<11 THEN GOTO 6070
6060 IF A(Z+Y)=H AND A(Z+2*Y)=E
THEN GOTO 6100
6070 LET Y=-9*(Y=-11)+9*(Y=-9)+1
1*(Y=9)+(Y=100)
6080 IF Y<>0 THEN GOTO 6055
6085 NEXT Z
6090 GOTO 6200
6100 LET A(Z)=E
6110 LET A(Z+Y)=E
6120 LET A(Z+2*Y)=C
6130 LET COMP=COMP+1
6140 RETURN
```

Program Five: after adding on line 60 as outlined in the text, you are ready to give the ZX intelligence! This program allows your computer to work out possible 'captures' and action them.

numbers of the squares which will be assigned. H$ holds the starting human squares, C$ the starting computer squares, B$, the empty or 'black' squares (they are white on our numbering diagram, but are black here to give a good appearance when the game is underway), and E$ for squares which will be empty at the start of the game, but which will be used for playing on once the game gets underway.

The first routine after the 'data' statements, lines 9050, 9060 and 9070, give a value of 9 to all squares. This value will later serve as an indication of 'off the board'. The lines from 9080 to 9100 give the values which will be assigned to the other squares. The variables are given names to make it easy to keep track of them during the game — H for human's piece, C for the computer's, E for an empty square and B for a black one.

Having run program two we need to check that it is working correctly, by printing out the board and seeing it is correct. Note that the RETURN line is numbered 9500 to give as much room as needed for working. Enter your program up to the end of program two, and make sure it runs through without a hitch. The code 9/8990 shows it is working perfectly. We will put the subroutine to print the board starting from line 8000. Add 30 GOSUB 8000, and then add program three, and run the whole program again.

If all goes well, a complete board, set up for CORNER CHECKERS, should appear. Once it has (and it is very pleasing to see the board on the screen as it looks far stronger than the printout would suggest), add 8130 RETURN, and you're ready to add the next part of the game.

Human Mover

The human moves are the simplest to program. In essence, all we need is an input to take the square the human is moving from, an input for the square the human is moving to, and a means of turning the 'square from' blank (E) and the 'square to' into a human square (H). It is also useful to check that the human is not cheating, and there will have to be some mechanism for 'erasing' pieces which the computer has jumped over, but — for now — let's just arrange for a simple, non-capture move. We'll start the PLAYER MOVE subroutine at line 7000. Add 40 GOSUB 7000, 50 GOSUB 8000, then enter program four.

Run this, and enter your move as suggested as 3344 — that is, two numbers. If all is well, you'll see the "H" move from square 33 to square 44. The program will keep cycling in its present form. Try moving a few other pieces, even com-

puter pieces. You'll see there is one check, line 7030, to make sure the move consists of four numbers.

This program includes a line to remove a piece which has been captured. Look at our master numbered board. If the player moves from 42 to 64, and there is a computer piece on 53, the piece on 53 must be removed. Fifty-three is half of 42 plus 64, which gives us an easy way of finding out which piece to 'delete'. Try out some 'captures', making sure the captured piece vanishes, and the human score is incremented.

Once you're happy with this, we start the biggest task of all, adding 'computer intelligence'. We'll start the computer's thinking subroutine at line 6000, so add 60 GOSUB 6000.

Computer Mover

Let's think about how the computer can be 'taught to play'. It must first scan the board, square by square, looking for any and all possible captures, so obviously it needs a loop of some kind. Look at lines 6010 to 6040 in program five. They go through the board, square by square, looking for a peice and once one is found, goes to line 6050 to find out what to do with that piece. The relationship between the squares on the board is plus eleven and minus eleven, plus nine and minus nine. The computer knows that if a human piece is on, say, a square eleven more than it is, and the square beyone that (its square number plus two times eleven) is empty, it can capture by jumping into the empty square. Add the lines between 6000 and 6200 (program five) and set up a capture or two for the computer, by moving some of your pieces into danger. It is fascinating (and quite pleasing) to see the computer finding possible captures, and making them. Random moves are the next thing we should implement.

We add the lines from 6200



onwards (program six) to achieve this. There are a number of things we need to do for 'intelligent' (ie. non-sacrifice) moves: find a piece (line 6230) and then check around this piece (from line 6260) to make sure that we are not moving the piece into a potential capture situation. If, after 200 squares have been chosen in this way, no move can be found, the computer goes to 6500 and choses 200 more squares, this time not checking to see if it is moving into danger. If no moves can be found, it goes to line 6500 and concedes defeat. Following through the possible moves will show you how this (somewhat complex) routine works.

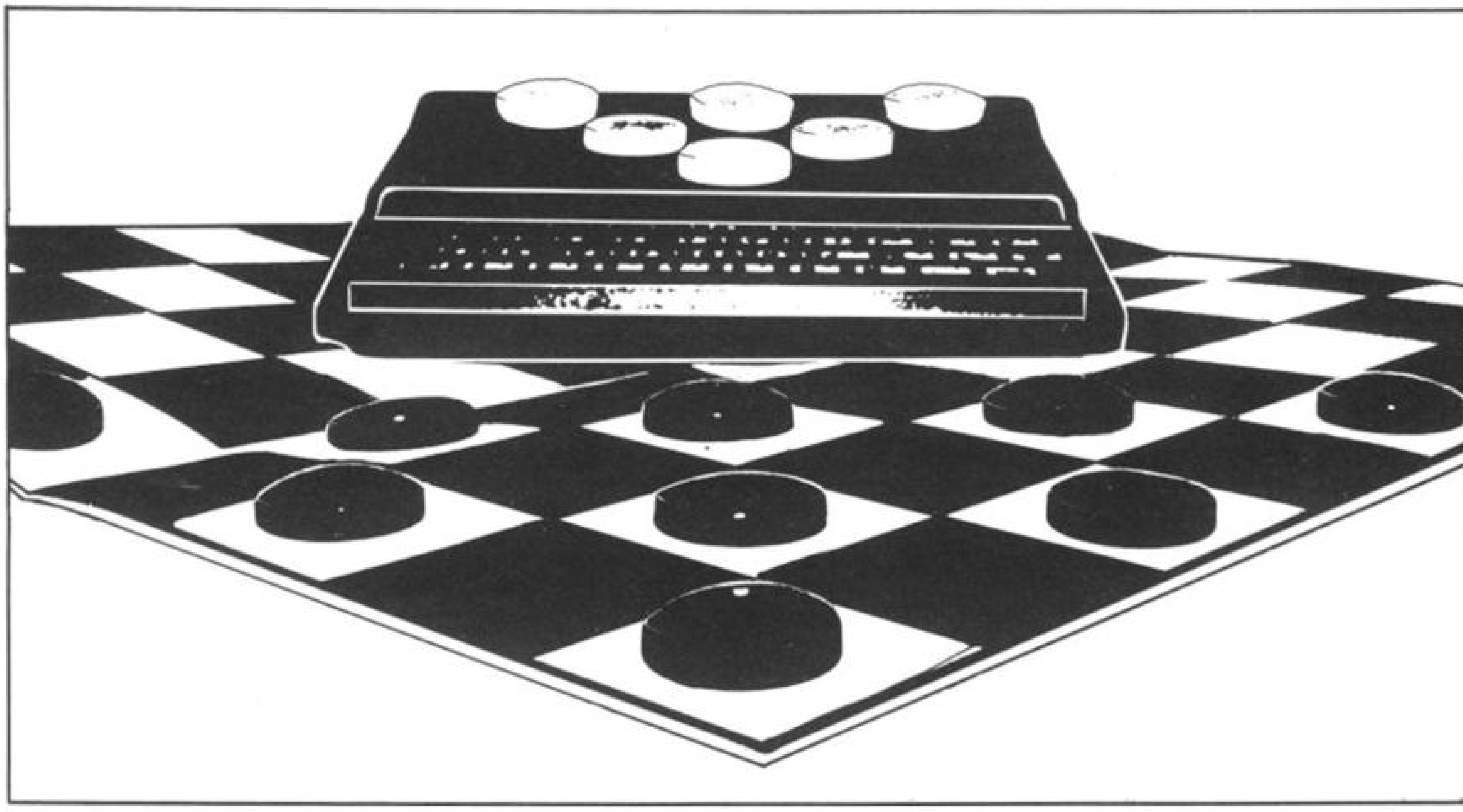
Finally, add the remaining lines for the main loop (program seven), which keeps the whole song and dance underway. The game continues until one of the player's manages to capture seven of the opponent's pieces.

There is quite a bit you can do to improve this game, including speeding it up by ensuring that when it picks squares at random it does not select the same one more than once in any particular move. Also, the printing of the board (especially the numbers down the left hand side of the board) slows the game down. You may well be ablve to improve this.

The final page of this article shows some stages in a typical game against this program.

In the next issue of ZX Computing, Tim Hartnell will look at a way of modifying this program to get a display which fills the entire screen including playing pieces which you define yourself. As well, he will show how 'Corner Chekcers' can easily be changed into 'Spanish Checkers'. A way to change the board subroutine so only the moved piece is reprinted will also be discussed.

Tim Hartnell is particularly interested in examining variations of the program given in this article, and in looking at programs derived from it. The best ideas (send complete description, and printer listing please) will be used in the next bumper issue of ZX Computing.



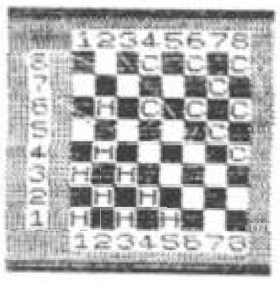
```
6200 REM **NON-CAPTURE MOVE**
6210 FOR Z=1 TO 200
6220 LET K=INT (RND*78)+11
6230 IF A(K)=C THEN GOTO 6260
6240 NEXT Z
6250 GOTO 6500
6260 LET Y=-11
6280 IF A(K+Y)=E THEN GOTO 6330
6290 LET Y=-9*(Y=-11)+9*(Y=-9)+1
1*(Y=9)+(Y=100)
6300 IF Y<>0 THEN GOTO 6270
6310 NEXT Z
6320 GOTO 6500
6330 IF K+2*Y>88 OR K+2*Y<11 THE
N GOTO 6400
6340 IF A(K+2*Y)=H THEN GOTO 624
0
6350 IF K-2*Y<11 OR K-2*Y>88 THE
N GOTO 6400
6360 IF A(K-2*Y)=H THEN GOTO 624
0
6400 LET A(K+Y)=A(K)
6410 LET A(K)=E
6420 RETURN
6500 FOR G=1 TO 200
6510 LET K=INT (RND*78)+11
6520 IF A(K)=C THEN GOTO 6600
6530 NEXT G
6540 PRINT AT 0,0;"I CONCEDE THE
 GAME"
6550 STOP
6600 IF A(K-11)=E THEN LET Y=-11
6610 IF A(K-11)=E THEN GOTO 6400
6620 IF A(K-9)=E THEN LET Y=-9
6630 IF A(K-9)=E THEN GOTO 6400
6640 GOTO 6540
6900 RETURN
```

Program Six: when no pieces are being taken, the move involved is simpler. It is, however, desirable to have the computer make a 'random' move occasionally so that games are not predictable and boring! This subroutine will throw in an occasional 'wild-cat' that just might *be brilliant!*

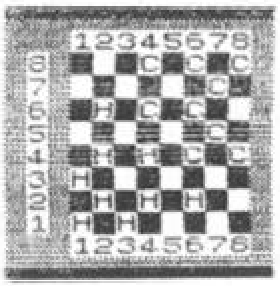
```
10 REM CORNER CHECKERS
20 GOSUB 9000
30 GOSUB 8000
40 GOSUB 7000
50 GOSUB 8000
60 GOSUB 6000
70 GOTO 30
```

Program Seven: after all the additions your introductory list should look like this. Each GOSUB will shift the action to a different section of the program in turn, producing a full interactive game.

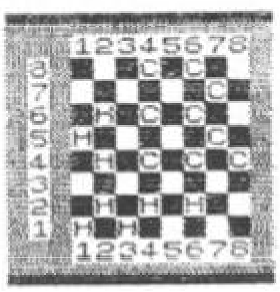
COMPUTER > 0 0 < HUMAN



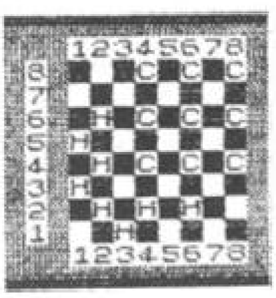
COMPUTER > 0 0 < HUMAN



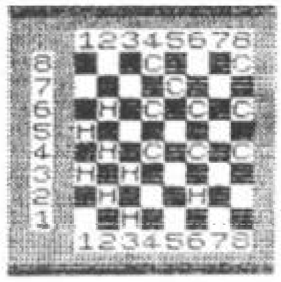
COMPUTER > 1 0 < HUMAN



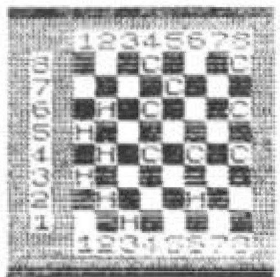
COMPUTER > 1 0 < HUMAN



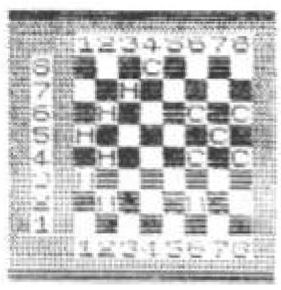
COMPUTER > 1 0 < HUMAN



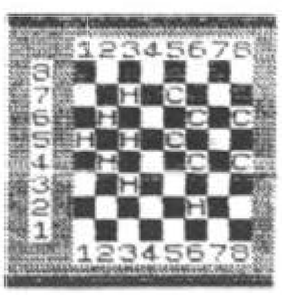
COMPUTER > 2 1 < HUMAN



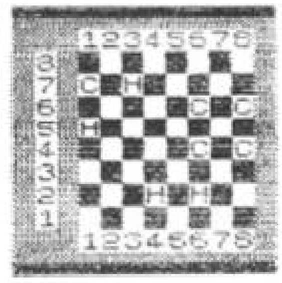
COMPUTER > 2 3 < HUMAN



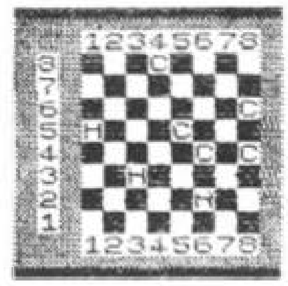
COMPUTER > 2 3 < HUMAN



COMPUTER > 5 4 < HUMAN

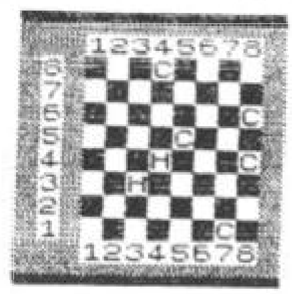


COMPUTER > 6 4 < HUMAN



I WIN

COMPUTER > 7 5 < HUMAN

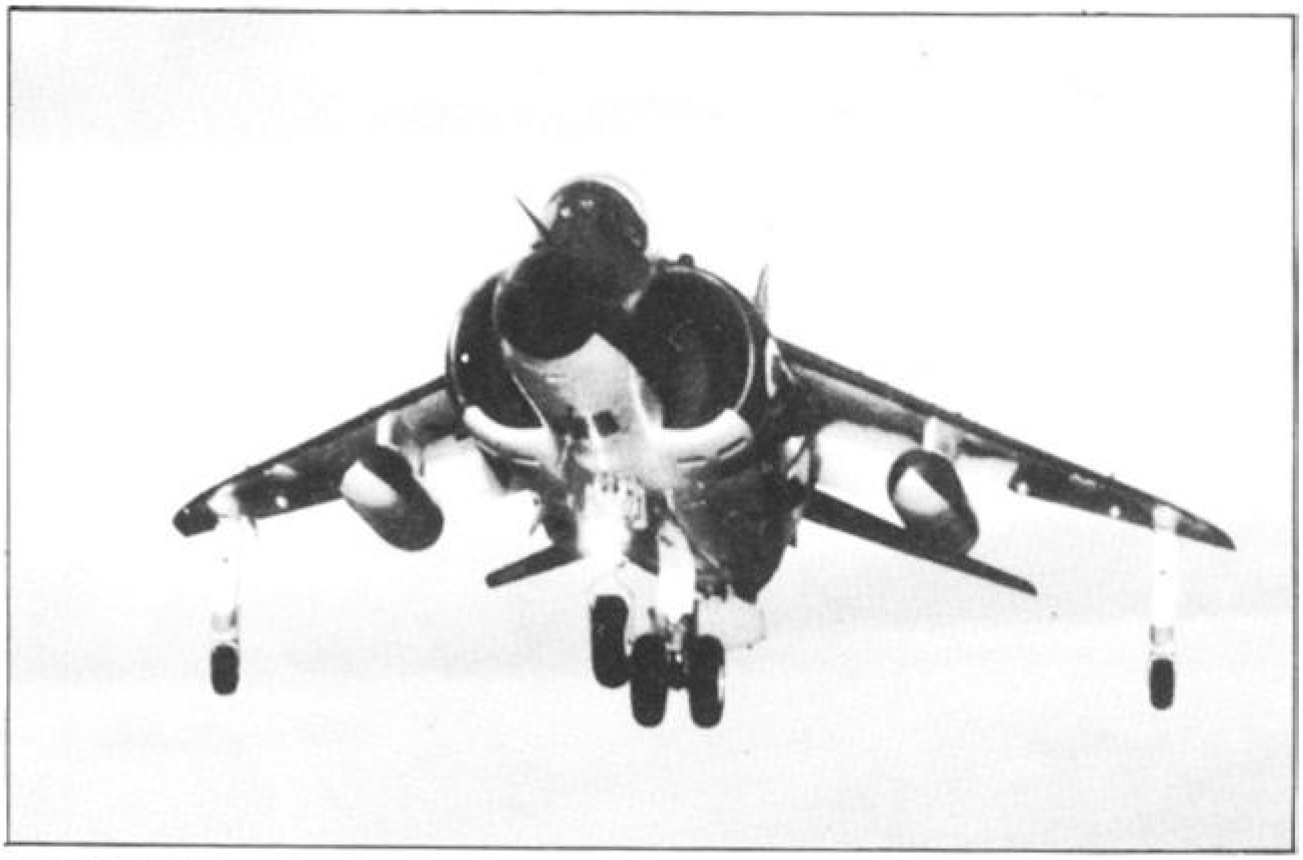


So you think you're smart, eh? Try and beat the program before you boast too much! Follow the game through (down each column in turn) and see if you would have done any better than our hapless editor who, although he got his nose briefly in front, ended up wiped out by the ZX!

The computer move routine is surprisingly effective in games like this where the moves are strictly limited in type and a logical approach is best most of the time. Your best chance of victory is to wait for a 'mistake' ie. a bad random move and then capitalise on it. If the ZX throws in a genius move instead . . . forget it!

Hit the deck

John Butler and Dave Groombridge have written this program in which you have to try and land a plane on an aircraft carrier, during a particuarly violent wind.



The screen display shows an aeroplane in the top left hand corner, with the sea at the bottom, and an aircraft carrier landing platform in the right hand corner.

The idea is to land your ZX80 aircraft onto the carrier's deck, taking note of the very gusty wind conditions given to you by the carrier's air traffic control.

These reports are given by a readout at the bottom of the screen, showing either 'E WIND' (easterly), 'W WIND' (westerly), or 'NIL WIND'.

If a westerly wind is prevailing, you will have to enter less power (power values are zero to 200), and the opposite for an easterly wind, when you'll need to enter more power. As a hint, the ideal landing speed (power input) is around 80 knots.

If you are too high, and unfortunately overshoot, press any key to start your approach again.

This program fits a 1K ZX80.

```
150 DIM A(17)
160 FOR I=0 TO 17
170 PRINT I,
180 INPUT X
190 LET A(I)=X
200 PRINT A(I)
240 NEXT I
```

PRESS RUN AND ENTER NEW LINE
(This will produce 0 in the top left hand corner of the screen. Enter the following values in turn, pressing NEW LINE between each value.)
0, 3, 5, 134, 4, 7, 0, 128, 133, 132, 128, 130, 0, 0, 3, 133, 0, 5,
PRESS NEW LINE
(The above lines will be over-written in the following listing:)

```
 10 LET V=700
 20 LET L=2000
 30 LET K=L/100
 40 LET D=1
 50 CLS
 60 IF K< 3 THEN GOTO 550
 70 IF D=4 AND K>4 THEN PRINT "OVER-SHOOT"
 80 LET J=20
 90 PRINT
100 IF J=K THEN GOTO 130
110 LET J=J-1
120 GOTO 90
130 IF D>3 THEN GOSUB V
140 IF D>2 THEN GOSUB V
150 IF D>1 THEN GOSUB V
160 FOR I=0 TO 2
170 FOR X=0 TO 5
189 PRINT CHR$ (A(X + 6*1));
190 NEXT X
200 PRINT
210 IF D>3 THEN GOSUB V
220 IF D>2 THEN GOSUB V
230 IF D>1 THEN GOSUB V
240 NEXT I
245 IF D=4 AND K>4 THEN GOTO 900
250 LET F=K
260 IF F=3 THEN GOTO 310
270 LET F=F-1
280 IF K=4 AND D=4 THEN GOTO 520
290 PRINT
300 GOTO 260
310 FOR Z=0 TO 13
320 PRINT "(shift A shift T)";
330 NEXT Z
340 PRINT "(shift S four times)"
350 LET W=RND (30)-RND (30)
360 IF W>0 THEN PRINT "W WIND ":W:"*****PWR 0 TO
    200?";
370 IF W<0 THEN PRINT "E WIND
    ";W+(2*W*-1);"*****PWR 0 TO 200?";
380 IF W=0 THEN PRINT "NIL WIND*******PWR 0 TO
    200?";
390 FOR P=0 TO 200
400 INPUT P
410 LET P=P+W
420 IF P<70 THEN GOTO 440
430 LET D=D+1
440 LET A=22
450 LET Q=P
460 LET Q=Q-10
470 IF Q<0 THEN GOTO 500
480 LET A=A-2
490 GOTO 460
```

```
500 LET K=K-A
510 GOTO 50
520 PRINT "(shift S six times)"
530 PRINT "LANDED";
540 STOP
550 PRINT "CRASHED"
700 PRINT,;
710 RETURN
900 INPUT H$
910 IF H$="" THEN GOTO 1
```

DO NOT PRESS RUN! (This will clear all the variables)
GO TO 1 NEW LINE

Pegging about





Havant, Hants, is the haunt of Michael Whitcombe, where his ZX80 has cooked up this program for the game of peg solitaire.

Fitting an unexpanded ZX80, the program holds the board as an array of seven elements, with each board position defined by two bits. Therefore, each array element defines a row of the board.

The first digit defines the vertical column (1-7), numbering from left to right, and the second digit defines the row (1-7), counting from the bottom to the top.

The first move will always therefore be 44. The program tests for the end of the game, although it cannot detect a stale-mate position.

```
10 DIM A(7)
20 LET A(1)=2016
30 LET A(2)=A(1)
40 LET A(3)=32766
50 LET A(4)=32638
60 LET A(5)=A(3)
70 LET A(6)=A(1)
80 LET A(7)=A(1)
90 FOR B-1 TO 7
100 LET C=2**(2*(8-B)-1)
110 FOR D=1 TO 7
120 LET E=(A(D) AND (3*C))
130 IF E=C*3 THEN PRINT " (shift A) * ";
140 IF E=2*C THEN PRINT " O * ";
150 IF E=0 THEN PRINT " * * ";
160 NEXT D
170 PRINT
180 PRINT
190 NEXT B
200 FOR B=1 TO 7
210 IF B=4 THEN GOTO 240
220 IF NOT (A(B) AND 10922)=0 THEN GOTO 280
230 GOTO 250
240 IF NOT A(B)=21972 THEN GOTO 280
250 NEXT B
260 PRINT " YOU WIN "
270 STOP
280 PRINT " YOU MOVE FROM * ";
290 INPUT B
300 IF B>75 OR B<13 THEN GOTO 290
310 PRINT B, " TO "
320 INPUT D
330 IF D>75 OR D<13 THEN GOTO 320
340 IF NOT (ABS(B-D)=2 OR ABS(B-D)=20) THEN GOTO
    470
350 LET C=B/10
360 LET E=D/10
370 LET F=(C+E)/2
380 LET G=2**((B+D-20*F)-1)
390 LET B=2**((B-10*C)*2-1)
400 LET D=2**((D-10*E)+2-1)
410 IF NOT (A(C) AND B)=B OR NOT (A)(E) AND 3*D)=2D
    OR NOT (A(F) AND G)=G THEN GOTO 470
420 LET A(C)=A(C)-B
430 LET A(E)=A(E)+D
440 LET A(F)=A(F)-G
450 CLS
460 GOTO 90
470 CLS
480 PRINT "INVALID"
490 GOTO 90
```

from

MIKRO-GEN the leaders in ZX games

Scramble



The high-speed arcade game. Easily the fastest available. 32 zones, thrust and altitude controls.

Supplied on cassette with library case

ONLY £3.95

Other great ZX games and add-ons from Mikro-Gen:

Space Invaders
the best you can get, with ever-increasing rate of play

Breakout
with seven bat angles to make it really difficult

Bomber
positively addictive

ZX Chess
The original – and still the best!

Sorcerer's Castle
takes you into a world of magical adventure

Above games all supplied on cassette with library case

£3.95 each (ZX Chess £6.50)

A/D Converter Board
Lets you connect analogue joysticks to the ZX81 digital inputs. Suitable for many other applications, easy to connect – and it improves RAM pack stability!
ONLY £18.50

Joysticks
Connect via our A/D Board – makes your ZX81 a true programmable games machine
ONLY £9.50 EACH

Disassembler/Monitor
An absolute must when learning machine code – lets you enter and run your own code
ONLY £3.95

Write for full details of the Mikro-Gen range of programs and add-ons, available from local stockists or direct from the manufacturers (please make cheques/PO's payable to Mikro-Gen and add 40p post & packing)

Suppliers of Software to Sinclair

MIKRO-GEN

24 Agar Crescent Bracknell Berks RG12 2BK
Tel: Bracknell (0344) 27317

ZX81 & 16K

THE TOMB OF DRACULA!



3D HORROR ADVENTURE GAME!

Occupying over 13½K of memory, a superb 3D graphics adventure game for the ZX81 with 16K RAM, **for only £3.95! Enter** Dracula's tomb at 30 minutes to sunsent ... **wander** through the tomb's pre-mapped 300 vaults in search of the fabled Vampire's Treasure ... **pick up** valuable silver stakes and use them to defend yourself against the lurking horrors ... ghouls, zombies, pits of primaeval slime ... **See** them all on the ZX81's plan of the tomb ... when it will let you! **Take** a chance on a Mystery Vault ... if your dare! And all the time the minutes are ticking by to sunset ... when **Dracula** rises from his coffin and comes after **you**! EaCH of the infinite levels of the tomb has its own 300 vaults ... go as deep as you like, the Prince of Darnkess will seek you out in his blood-lust! **WARNING:** people of an exceptionally nervous disposition should play this game only during the hours of daylight! Special facility enables a game in progress to be saved on tape so you can continue it whenever your choose.

Price of only **£3.95** includes ready-to-load cassette with library case and inlay, full instructions, postage and packing. Order today! Money refunded if not delighted! Send cash, P.O. or cheque to:

MOVIEDROME VIDEO DEPT. ZC1
19 Leighton Avenue, Pinner, HA5 3BW.

NOW!

Spectrum workstation



Following the success of our ZX81 plinth we have introduced a workstation for the Spectrum. This stylish ABS plinth raises and tilts the TV for better viewing whilst angling the Spectrum and making typing easier. The PSU is hidden underneath, the printer and cassette may still be used, and a matching, stackable unit for Microdrives will be available.

Peter Furlong Products, Unit 5, South Coast Road Industrial Estate, Peacehaven, Sussex BN9 8NA. Tel. (07914) 81637.

Getting a little joy

Jeremy Ruston takes a firm grip on a Microgen joystick . . . and likes what he finds.

The Microgen joystick system costs £19.80 for the controller board, and £9.60 for each joystick – a maximum of two can be attached.

The controller board fits between the ZX81 edge connector and the 16K RAM pack. This arrangement may look a little unwieldy, but I am assured that due to the fact that the RAM pack is now hanging at a slightly different angle, it is fully supported by the table, which apparently clears up the lingering problems with the pack.

The side of the board, which is not cased, has two sockets for the two possible joysticks. The standard of construction of the board is very high, although sockets have not been used for the nine integrated circuits – which if anything makes the connections more reliable. Besides the ICs, there are 20 or so discrete components, and a potentiometer on the board. The potentiometer is used to adjust the range of values generated by the joystick. This only needs to be done once, and Microgen supply a short program to assist in setting this potentiometer.

Making up for X

The joysticks themselves are made by Radio Shack (Tandy), and look rather like a RAM pack with a pencil sticking through it. The stick itself does not return to the central position when it is released, but it's very subjective if you like it to or not.

The cable supplied is good and long, so even if your ZX81 is forced to sit within two feet of your TV by Sinclair's idea of an aerial lead, you can play games from a decent distance, in comfort.

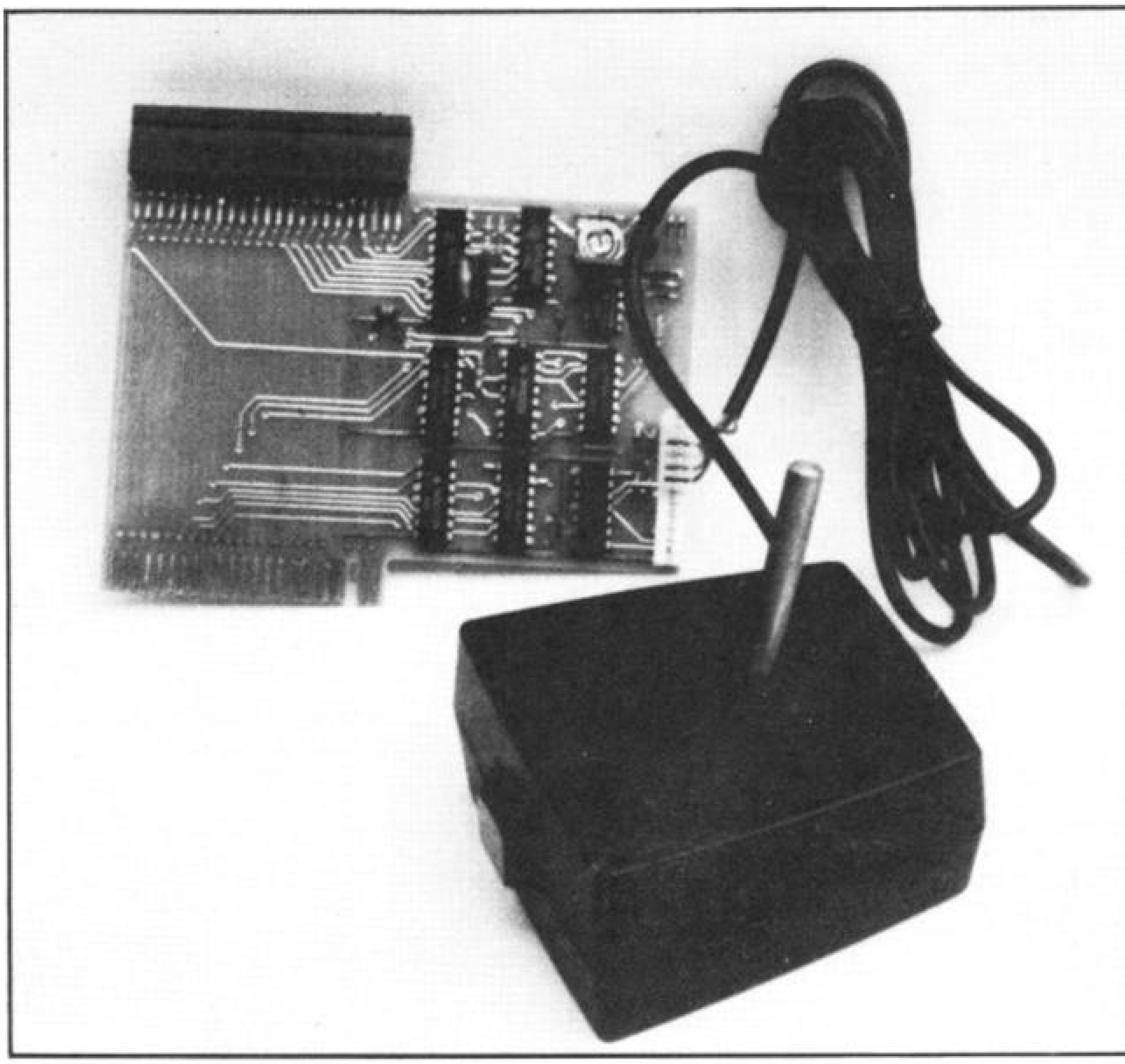
So much for the hardware, but what of the software? All data transfers between the computer and the joysticks are made by PEEKing and POKEing to and from location 16000 (decimal). Before reading data from the joystick it is necessary to POKE a number to the location, to specify whether you wish to read Joystick 1 or 2, and whether you wish to read the X or Y values. That is basically all there is to using them, except that if a value greater than 128 is read, the 'fire' button (provided on each stick) is being pressed. I found it very easy to write simple programs using these devices, even in machine code, and any game written in BASIC should be easy to convert.

In Conclusion

To sum up, I feel these make interesting peripherals for the ZX81 (and probably ZX80) enthusiast, but – and it's a big but – the system does come out to be expensive, in relation to a ZX81 in kit form. Thoroughly recommended.

As a postscript, I've just heard that the Microgen boffins are developing a range of quality games (about a fiver a cassette) to use the joystick, including Space Invaders. I was supplied with a maze game, which they at Microgen called 'a side-B game', which was fairly impressive.

Many thanks to Microgen for the review Joysticks.



A test of skil

```
   5 REM "GRAND PRIX"
  10 PRINT "  ** THE GRAND PRIX P
ROGRAM **"
  20 PRINT ,,,,"USE THE KEYS Q W
 E";TAB 13;"A   D";TAB 13;"Z X C
";TAB 8;"TO STEER YOURSELF,","KE
YS N,M TO BRAKE/ACCELERATE"
  25 PRINT ,,"WARNING: PRESSING
ANY OTHER KEY","  WILL STOP THE
CAR"
  30 PRINT ,,"HOW MANY LAPS? ";
  40 INPUT L1
  45 LET L=1
  50 PRINT L1
  54 DIM A$(20,32)
  55 DIM B$(20,50)
  57 FAST
  60 GOSUB 1000
  65 CLS
  68 LET YL=0
  70 FOR X=1 TO 20
  80 LET Z=1
  90 IF B$(X,Z)=" " THEN GOTO 14
0
 100 LET Y=VAL B$(X,Z TO Z+1)
 110 PRINT AT X,Y-1;"*"
 115 LET A$(X,Y)="*"
 120 LET Z=Z+2
 130 GOTO 90
 140 NEXT X
 160 LET S=0
 165 LET T=0
 170 LET U=100
 175 LET A$(14,26)="-"
 180 SLOW
 190 LET X=14
 200 LET Y=26
 210 PRINT AT X,Y-1;"■"
 220 PRINT AT 0,0;"READY..";
 230 FOR W=1 TO 3
 240 PRINT 4-W;"..";
 250 PAUSE 50
 260 NEXT W
 265 PRINT AT 0,0;"SPEED:0      TI
ME:0       LAP:1"
 270 LET F$="X"
 280 GOSUB 1300
 290 PRINT AT X,Y-1;A$(X,Y)
 300 LET X=X+I
 310 LET Y=Y+J
 320 PRINT AT X,Y-1;"■"
 322 IF A$(X,Y)<>" " THEN GOSUB
1500
 324 LET T=T+1+U/100
 325 PAUSE U
 327 LET S=2*(100-U)
 328 PRINT AT 0,6;S;"  ";AT 0,16
;T;"  "
 330 IF INKEY$="" THEN GOTO 290
 335 IF INKEY$="M" OR INKEY$="N"
 THEN GOTO 1400
 340 LET F$=INKEY$
 350 GOTO 280
 990 STOP
 999 REM COURSE DATA
1000 LET B$(1)="12131415"
1010 LET B$(2)="101116252627"
1020 LET B$(3)="0708091314172428
"

1030 LET B$(4)="0611121518222326
29"
1040 LET B$(5)="0405080910161920
21252730"
1050 LET B$(6)="0203071723242831
"

1060 LET B$(7)="0105061819202122
2931"
```

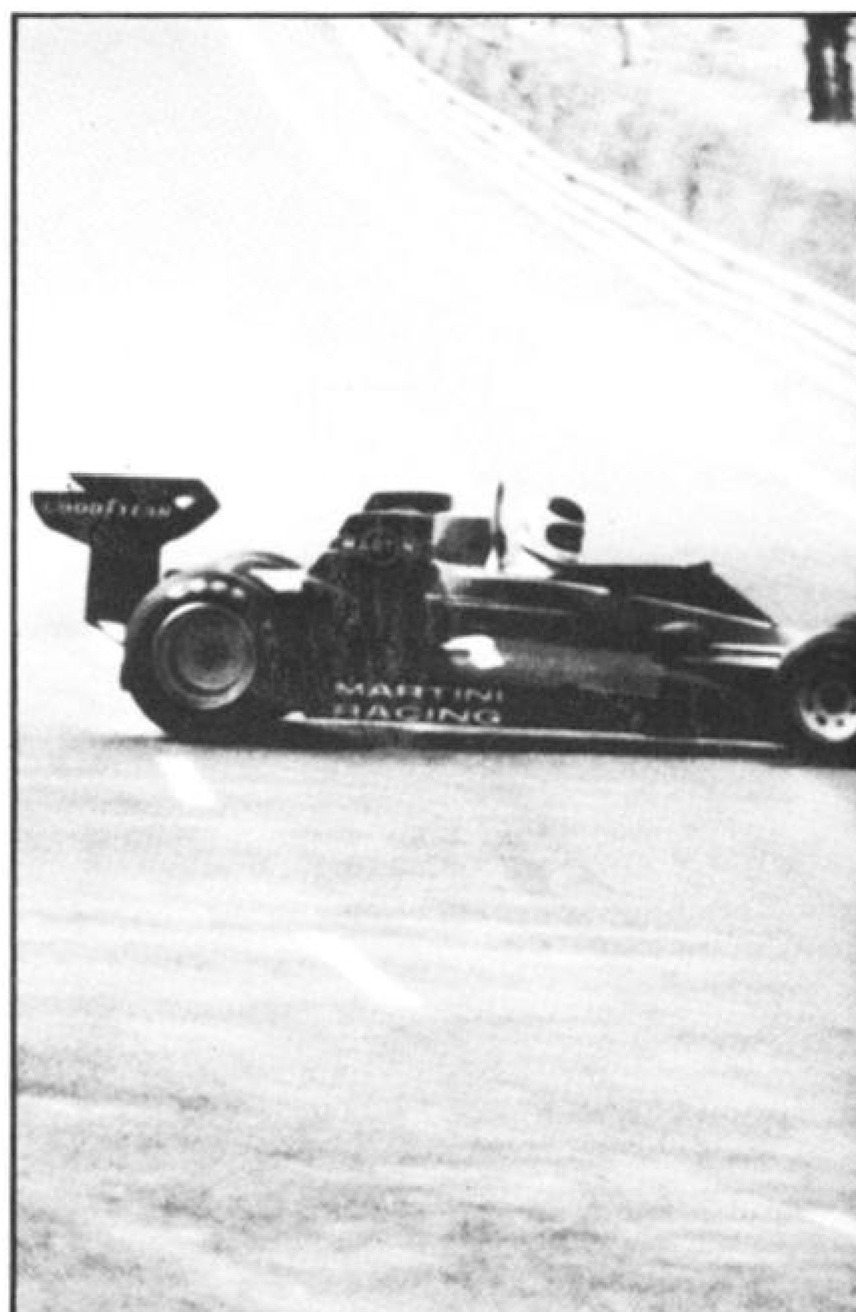
Jim Archer of Frimley, Surrey, puts you in command of your ZX81 in a well written GRAND PRIX program.

This is a game of skill, combining steering, braking and accelerating around a rather complex race track.

The car is steered into a starting speed of 40 mph from which you can accelerate up to a maximum of 200 mph, but it is quite difficult to complete the course at this speed without crashing at least once, and every crash costs you an extra 10 seconds.

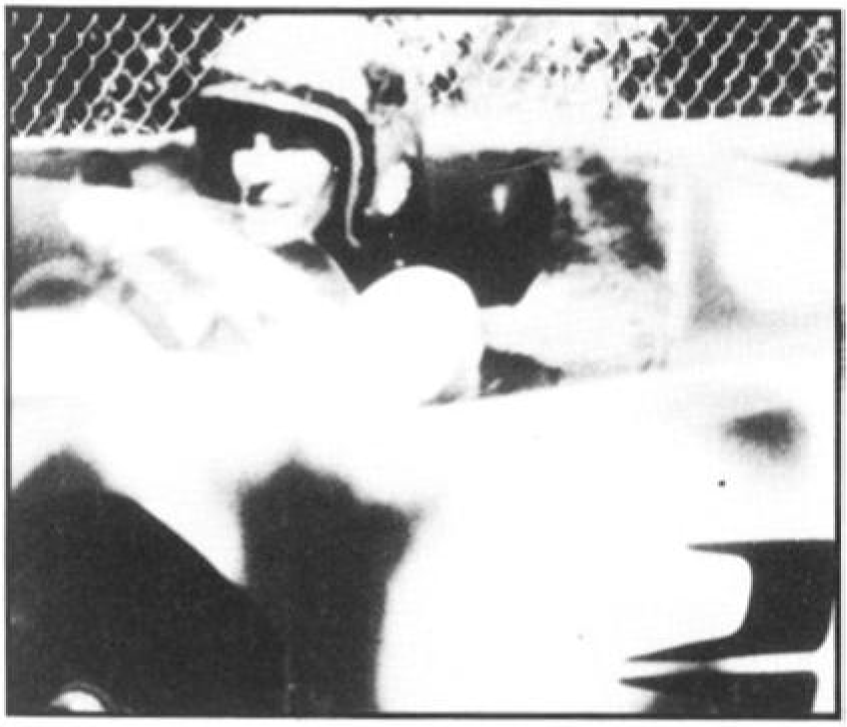
There is a PAUSE statement within the main loop which is related to the current speed, so the program does actually get faster as your speed goes up.

At the final lap, the average time/lap is given in minutes and seconds and the driver is graded between "A" and "F". Only the best can attain an "A" – the "CONCEIT" grade.



and nerves!

ement
ich is
ed, so
ly get
up.
erage
s and
raded
ly the
– the

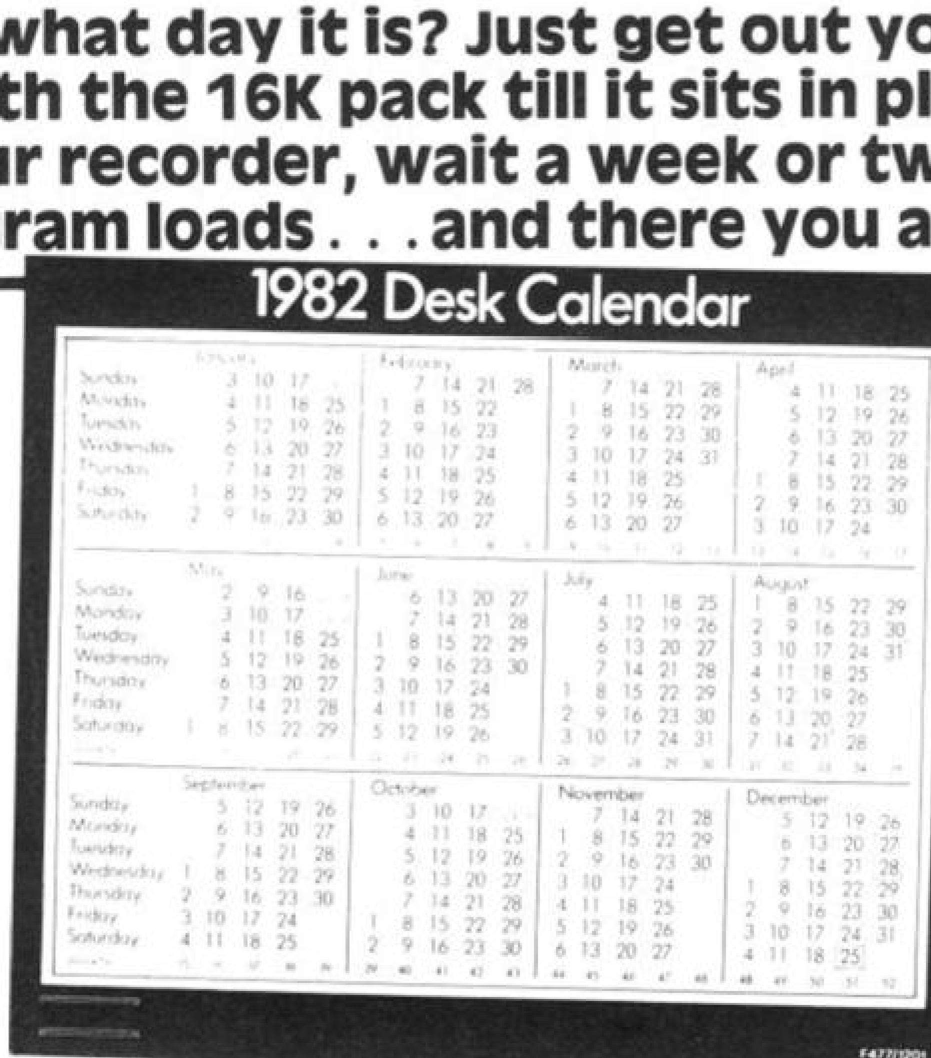




```
1070 LET B$(8)="01041213141516291
31"
1080 LET B$(9)="020411172932"
1090 LET B$(10)="020508091011131
415183032"
1100 LET B$(11)="030507131618303
2"
1110 LET B$(12)="030507091011121
5161820212223242526272830 32"
1120 LET B$(13)="030507101418192
93032"
1130 LET B$(14)="020506101416171
8212223242526272930 32"
1140 LET B$(15)="010407061014202
7293032"
1150 LET B$(16)="010306101516171
8192223242526272930 32"
1160 LET B$(17)="010406080921293
032"
1170 LET B$(18)="020506082123242
52627282930 32"
1180 LET B$(19)="03082132"
1190 LET B$(20)="040506072223242
5262728293031"
1195 RETURN
1300 LET I=(F$="Z")+(F$="X")+(F$
="C")-(F$="Q")-(F$="W")-(F$="E")
1310 LET J=(F$="E")+(F$="D")+(F$
="C")-(F$="Q")-(F$="A")-(F$="Z")
1320 IF F$<>"" AND U=100 THEN LE
T U=80
1330 IF I=0 AND J=0 THEN LET U=1
00
1340 RETURN
1400 IF INKEY$="M" AND U>=20 THE
N LET U=U-20
1410 IF INKEY$="N" AND U<=80 THE
N LET U=U+20
1430 GOTO 290
1500 IF A$(X,Y)="-" THEN GOTO 16
00
1505 LET U=100
1510 LET T=T+10
1515 IF S=0 THEN RETURN
1520 PRINT AT 21,5;"***CRASH***"
1530 PAUSE 60
1540 PRINT AT 21,5;"           "
1550 LET F$=""
1560 LET S=0
1570 LET I=0
1580 LET J=0
1590 RETURN
1600 IF S=0 THEN RETURN
1605 LET L=L+1
1607 IF L>L1 THEN GOTO 1630
1610 PRINT AT 0,27;L
1620 RETURN
1630 LET T=T/L1
1635 LET M=INT (T/60)
1640 LET S=INT (T-60*M+.5)
1650 PRINT AT 21,0;"AV/LAP:";M;"
:";
1660 IF S<10 THEN PRINT "0";
1670 PRINT S;" MIN:GRADE ";
1680 IF T<=105 THEN PRINT "A-CON
CEIT"
1690 IF T>105 AND T<=125 THEN PR
INT "B - FAST"
1700 IF T>125 AND T<=175 THEN PR
INT "C-AVERAGE"
1710 IF T>175 AND T<=200 THEN PR
INT "D-MEDIOCRE"
1720 IF T>200 AND T<=225 THEN PR
INT "E - SLOW"
1730 IF T>225 THEN PRINT "F - SN
AIL"
1750 STOP
```

Going Gregorian

Want to know what day it is? Just get out your ZX81, fiddle with the 16K pack till it sits in place, connect up your recorder, wait a week or two while this program loads . . . and there you are.



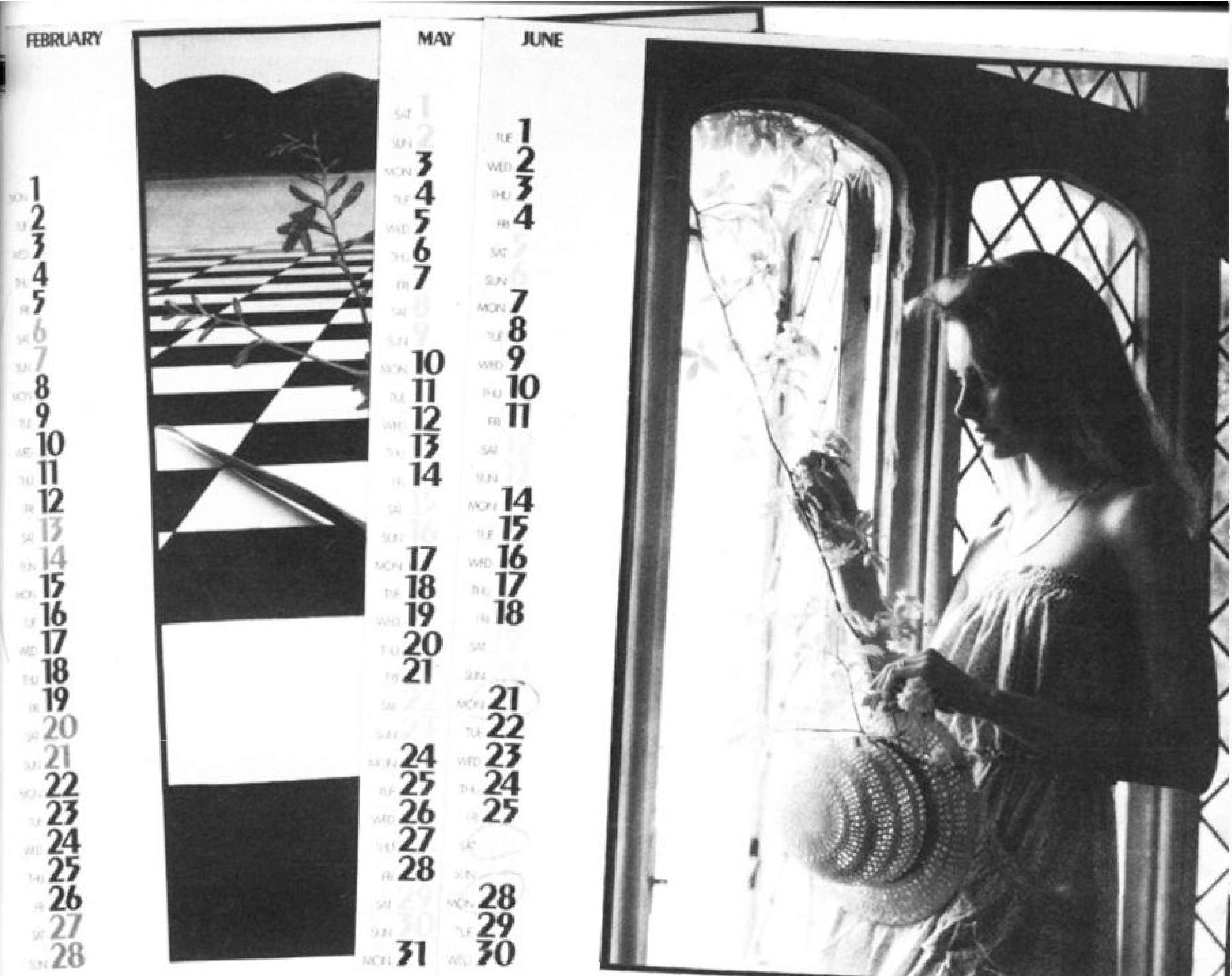
The program is designed to print out the calendar of any specified year, accurately and neatly, or else just a particular month of interest, or alternatively to state on which day of the week any date falls.

It will work for any date after 1752 (when 11 days were added to correct the Roman Calendar).

This program will be especially pleasing to people with a line printer to print out a wall calendar. Just use COPY for each month displayed.

***CALENDAR* was also written by Jim Archer.**

```
   5 REM "CALENDER"
   7 PRINT TAB 10;"CALENDER";AT
2,00;
  10 PRINT "FOR A YEARS CALENDER
 PRESS Y,",",FOR JUST ONE MONTH P
RESS M,",",OR TO FIND DAY OF WEEK
 PRESS D."
  15 LET S=1982
  20 IF INKEY$="" THEN GOTO 20
  25 LET W$=INKEY$
  27 PRINT
  30 IF W$="Y" THEN GOTO 50
  35 IF W$="M" THEN GOTO 1500
  40 IF W$="D" THEN GOTO 800
  45 PRINT "I BEG YOUR PARDON?"
  46 PAUSE 30
  48 GOTO 20
  50 PRINT "YEAR OF CALENDER?"
  55 LET R=0
  60 INPUT S
  65 GOSUB 1000
  68 FAST
  70 LET M$="JANUARY:31FEBRUARY:
28MARCH:31APRIL:30MAY:31JUNE:30J
ULY:31AUGUST:31SEPTEMBER:30OCTOB
ER:31NOVEMBER:30DECEMBER:31"
  75 LET J=0
  80 FOR P=1 TO 12
  90 LET N$=""
 100 LET J=J+1
 110 IF M$(J)=":" THEN GOTO 140
 120 LET N$=N$+M$(J)
 130 GOTO 100
 140 LET O=VAL (M$(J+1 TO J+2))
 150 LET J=J+2
 160 IF P=2 AND S/4=INT (S/4) AN
D (NOT S/100=INT (S/100) OR S/40
0=INT (S/400)) THEN LET O=29
 162 IF W$="M" AND P<R THEN GOTO
 300
 164 CLS
 165 IF LEN N$>4 THEN PRINT " ";
 166 PRINT "            ";S
 167 GOSUB 1050
 170 PRINT TAB 11;N$
 180 PRINT
 185 GOSUB 1050
 190 PRINT "   M   T   W   T   F
   S   S"
 200 PRINT
 205 LET D=0
 207 LET B=8-Q
 208 IF B=7 THEN LET B=0
 210 PRINT TAB (4*B+1);
 220 PRINT "  ";
 230 LET D=D+1
 235 IF P=12 AND (D=25 OR D=26)
THEN GOTO 340
 240 PRINT D;
 250 LET C=D-Q+1
 260 IF C/7=INT (C/7) THEN GOSUB
 1050
 280 IF D<9 THEN PRINT " ";
 290 IF D<O THEN GOTO 220
 295 STOP
 300 LET W=(35+Q-O)/7
 310 LET Q=INT (W*7-(INT W)*7+.1)
 315 IF Q=0 THEN LET Q=7
 320 NEXT P
 325 SLOW
 330 GOTO 9999
 340 IF D=25 THEN PRINT "25";
 345 IF D=26 THEN PRINT "26";
 350 GOTO 250
```




```
 790 PRINT "YOURE PULLING MY LEG ","  NOW PUT IN A REAL DATE"
 800 PRINT "DATE? ";
 810 INPUT U
 815 PRINT U
 820 PRINT "MONTH?(NUMBER) ";
 830 INPUT P
 835 PRINT P
 837 IF P<1 OR P>12 OR U<1 OR U>31 OR P<>INT P THEN GOTO 790
 840 PRINT "YEAR? ";
 850 INPUT S
 855 PRINT S
 860 GOSUB 1000
 870 GOSUB 1100
 880 LET T=U+8-Q
 890 LET Y=INT (T-7*INT (T/7)+.1)
 900 LET M$="SUNDAY   MONDAY   TUESDAY  WEDNESDAYTHURSDAY FRIDAY   SATURDAY "
 910 LET A=9*Y+1
 920 LET Z$=M$(A TO A+8)
 925 GOSUB 1050
 930 PRINT "  ";U;"-";P;"-";S;" WAS A ";Z$
 935 GOSUB 1050
 940 PRINT "ANY MORE?"
 950 INPUT W$
 960 IF W$="Y" THEN GOTO 800
 970 GOTO 9999
1000 LET W=S-2001
1010 LET X=8-(W+INT (W/4)-INT (W/100)+INT (W/400))
1020 LET Q=INT (X-7*INT (X/7)+.1)
1030 IF Q=0 THEN LET Q=7
1040 RETURN
1050 PRINT
1060 PRINT
1070 RETURN
1100 IF P<3 THEN GOTO 1180
1110 IF P=3 OR P=11 THEN LET Q=Q+4
1120 IF P=4 OR P=7 THEN LET Q=Q+1
1130 IF P=5 THEN LET Q=Q+6
1140 IF P=6 THEN LET Q=Q+3
1150 IF P=8 THEN LET Q=Q+5
1160 IF P=9 OR P=12 THEN LET Q=Q+2
1170 IF S/4=INT (S/4) THEN GOTO 1200
1180 IF P=2 THEN LET Q=Q+4
1190 IF Q>7 THEN LET Q=Q-7
1195 RETURN
1200 LET Q=Q-1
1210 GOTO 1190
1490 PRINT ,"THE NUMBER OF A MONTH PLEASE"
1500 PRINT "MONTH TO BE PRINTED(NUMBER)? ";
1510 INPUT P
1515 PRINT P
1517 IF P<1 OR P>12 OR P<>INT P THEN GOTO 1490
1520 PRINT "OF THE YEAR? ";
1530 INPUT S
1535 PRINT S
1540 GOSUB 1000
1550 LET R=P
1560 GOTO 68

9999 STOP
```

Edinburgh rules O.K.

The Edinburgh ZX Computer Show, organised by Gordon Hewit and the Edinburgh ZX Users Club, was a resounding success.

Over 1500 people attended the one day show, held in the entrance foyer to a sports stadium, stretched over a quarter of a mile, meaning a long trek was needed to get from one end to the other.

Exhibitors at the fair included Logan Software, Haven Hardware, Artic Computing, Fuller Micro Systems, the local W H Smith store (which did a lot to help publicise the show, and appears a very good force in the city for the promotion of computer use), Richard Shepherd Software, JRS Software, Redditch Electronics, V&H Computer Services ('What Can I Do with 1K?') and Video Software Ltd.



Club members provided a continuous demonstration of the capabilities of the ZX computers at the Edinburgh ZX Computer Show



Show organiser, Gordon Hewit

Ian Logan at the show with the book he wrote with Dr Frank O'Hara: 'Sinclair ZX81: ROM Disassembly, Part B'

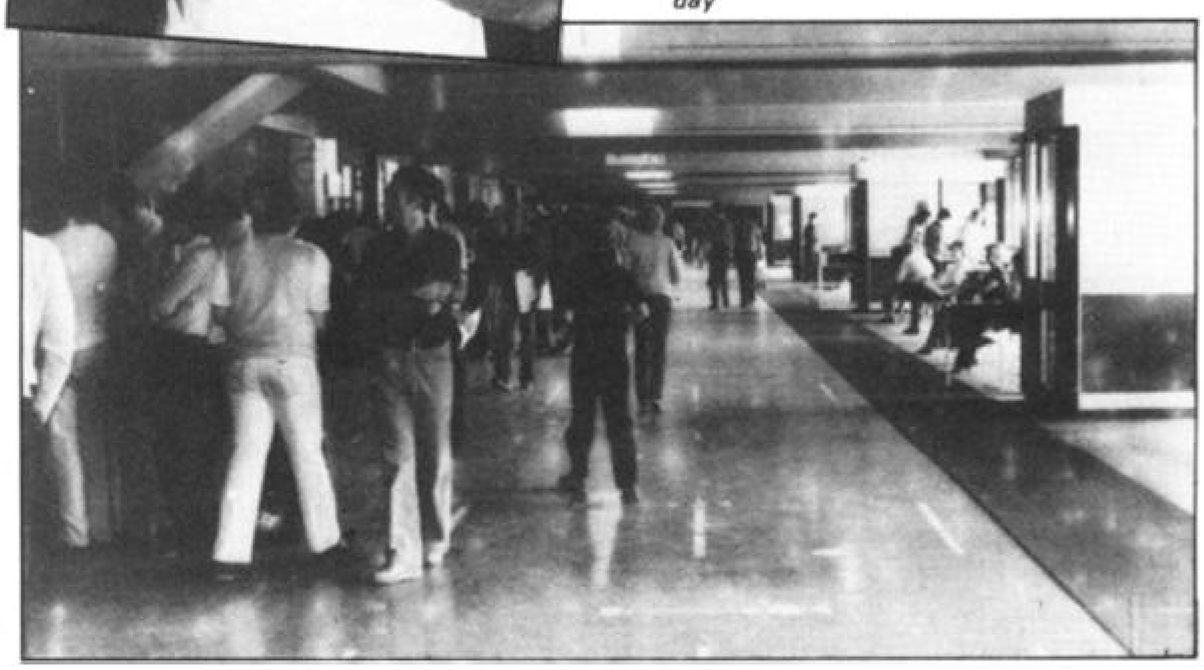




Alastair Gourlay, author of '34 Amazing Games for the 1K ZX81' came over from Glasgow for the day



Macronics demonstrated their ZX81 disc unit



Exhibitors were strung out along the concourse of a sports stadium

Getting Picturesque

Picturesque in West Wycombe have produced a machine code de-bug monitor tape known as the ZX-MC. Designed to run on the 16K ZX81 it allows you to enter, run and de-bug machine code programs independently of BASIC programs.

ZX-MC resides in RAM and leaves you with 12½K of memory to work with. You can SAVE and LOAD at double speed any specified area of free RAM. It means you don't have to store your machine code in ERASE or in REM lines. ZX-MC is supplied on cassette with a 32 page operating manual. The manual alone is superbly done and is an object lesson to other people producing products for the ZX81.

Picturesque also produce a screen kit which they call Screen Kit 1 which is a suite of machine code routines that are called from BASIC programs to enhance your screen display. You can do such things as drawing a border or clearing only part of the screen. There's a keyboard scan with flashing cursor; you can invert the video of part of the screen; it'll tell you how much memory is left; there's a routine to LOAD any character to the whole screen or clear the screen by strolling up, down, left or right. Screen Kit 1 also has a routine to allow you to create data files on cassette. It lets you SAVE and LOAD at double speed just the basic variables. Machine Code De-bug Monitor and Screen Kit 1 are available from Picturesque who are at 6 Corkscrew Hill, West Wycombe, Kent, BR4 9BB.



Vogue cover model, Australian Sue Currier, now heads her own ZX software company in New York, Softsync, Inc. Sue was in London looking for the best of British software to take back to the States to sell to Timex/Sinclair 1000 owners. Among packages she most admired was Bugbytes 'Mazogs', and software from Quicksilva. Softsync's address is P.O. Box 480, Murray Hill Station, New York, NY 10156, (212) 685-2080.

The Boston-based ZX-support company, Mindware, has also been over here several times looking for good ZX software and other products to sell in America. Mindware Inc., can be contracted at 70 Boston Post Road, Wayland, Massachusetts, (617) 358-7175. Both Mindware and Softsync are always on the lookout for good, new products which support the ZX computers and would welcome hearing from you if you have developed anything interesting. Both companies are signing rights deals to distribute the products in America.



Some of the highly effective ZX graphics suggested by Print 'n' Plotter



The Plot Thickens

If you want a simple way to make your ZX81 or Spectrum programs more exciting the products available from Print 'n' Plotter will help you.

Essentially they sell a pad called a "Print 'n' Plotter Jotter" which has a grid of squares for PRINT AT locations and much smaller ones for using the PLOT commands on the ZX81. All you do is fill in the squares with the picture that you finally want to get on to your screen and then transfer it across. As their graphics publication shows, some remarkably effective displays can be produced doing this.

There is also a Print 'n' Plotter film designed to be laid over a picture so you can trace over a picture before transferring it to the TV screen. The graphics booklet, which is available for £1.50, includes comprehensive instructions for making your programs look better and a complete listing for a rather exciting fruit machine program called Tutti Fruiti.

The booklet also gives advice on how to make programs which include a lot of numbers in the output more interesting. A way of scrolling test graphic displays is given and suggestions for graphic tape indexing.

The effectiveness of the ZX81 for graphic displays is perhaps best known when it is used to output to the printer rather than the TV screen. There is a very effective sample of a stuka bomber. The graphics programming book and the jotter and film are available from Print 'n' Plotter, 19 Borough High Street, London SE1.

What People are Saying

As can be imagined, a number of publications in America have taken note of the ZX81 and Clive Sinclair.

In March 'Fortune' magazine described Clive as 'a shy, baldish, 41 year old, famous until now as an electronics wizard who kept getting his entrepreneurial wires crossed'.

"Like the hula hoop, Sinclair's micro marvel, dubbed the ZX81, was an instant and overwhelming success... So popular has the machine proved that it has spawned over 150 new businesses from manufacturers of add-on hardware to publishers of fan magazines and software...". Nigel Searle, who was head of Sinclair Research in Boston Massachusetts, but is now in charge of the software division of Sinclair Research in England, told 'Fortune' magazine, "The Sinclair phenomenon is to enter those races that are worth winning but that no one else even knew were going on."

In May the American magazine 'Popular Computing', under the headline 'Big Power in a Small Package' said about the ZX81 "The innards of the ZX81 resemble an electronic watch... Despite obvious limitations the Sinclair ZX81 is well designed, very useful and a bargain... Although its current uses are limited to learning BASIC and a few small application programs the Z80A microprocessor gives the Sinclair the ability to achieve much more. By the end of the year there may be more ZX81's in the world than any other computer..."

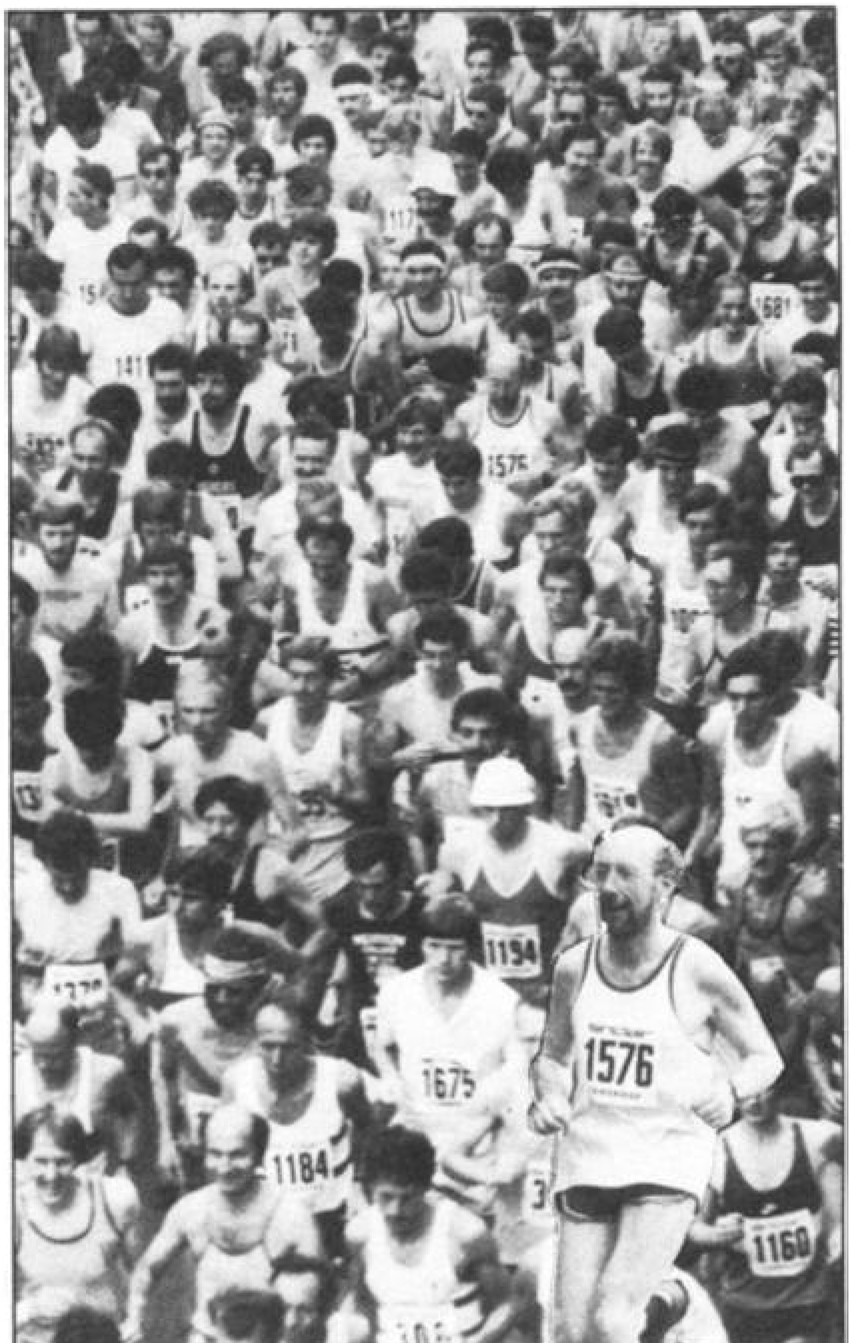
"Small size and a low price do not have to restrict a computer's capabilities." That was the headline for a review of the ZX81 which appeared in Radio Electronics in April "It has always seemed that when someone said that something couldn't be done Sinclair has set out to do it so it was only natural that he brought out a full featured 8 byte micro computer for under $200... When you first look at the ZX81 it looks like a very modest micro computer... yet its developer is quick to defend its capabilities.

When the new version of the ZX81 was introduced at a Boston press conference late last year Sinclair said it was "not a reduced support machine. The language it uses is complete". Radio Electronics concluded: "In the final analysis the ZX81 is a building block unit. It is intended to train people who know little or nothing about computers, that is all it is intended to be."

At the end of last year 'Business Week' which described the ZX81 as a dirt cheap personal computer, ran a story on Sinclair under the headline 'A British Computer Hits it Big'. The article quotes a Benjamin M Rosen, who it describes as an industry-watcher, as saying "Sinclair's got a tiger by the tail. It's not a question of price performance, it's only a question of price. Sinclair has found a segment of the market that no one else has found. It seems to be a big one."

Les Solomon, writing in the March issue of 'Popular Electronics' says: "We were surprised at the amount of computing power that Sinclair packed into such a small computer (you can carry this little wonder in a jacket pocket without making a bulge). The BASIC is as good as anything around in small computers and has commands that others do not have.. The 164 page manual is one of the best we have seen... There are many typically British expressions, but most readers will readily understand them...

"The Sinclair ZX81 looks like a winner for those who want a low cost way to learn BASIC programming or a small, inexpensive yet powerful computer to start out on... The old saw about good things coming in small packages is true in the case of the Sinclair ZX81..."



Clive (can you spot him in the crowd at the start?) Sinclair was sponsor of the Cambridge half-marathon during the Cambridge Festival. Every runner sported a number bearing the magic name. Clive, a veteran of several marathons (including the New York marathon, twice) ran the full distance, and arrived at the finishing line looking little the worse for wear.

Expressing an Interest

The response to this test-mailing was so good they decided to offer it to 2,000,000 of their card holders.

At the end of last year in America the American Express Credit Card mailed a selected number of their card holders with a catalogue which included the ZX81.

The response was immediate and they had 2,000 orders by noon the first day after the offer was made. Margaret Bruzelius, who is the head of Sinclair's operation in America, said that since the offer was made they'd continued to sell at nearly 2,000 a day.

The £49.95 ZX 81



The ZX81 price has been cut, as was expected, from £69.95 to £49.95.

As well as the price cut, Clive has decided to push the ZX81 through retail outlets in addition to W H Smith.

Two other retail deals have been announced, and discussions with others are said to be 'at an advanced stage'.

Boot and Greens, a subsidiary of Debenhams, will sell the ZX81 and the associated 'official' software, and Prism Microcomputers will sell the ZX81 as a wholesaler through high street computer shops.

This means that, for the first time, the ZX81 will be treated like nearly all other computers, and the numbers in use (believed to be around half a million) is sure to swell even more.

The ZX81 is being produced at a rate of 60,000 a month. It is not clear how many of those are destined to end up in America, miraculously transformed into 2K Timex/Sinclair 1000's.

Sinclair gets the nod

After being excluded from the BBC program and from the government's "Let's put a micro in every secondary school" scheme, no one could have blamed Clive and his gang for feeling that someone up there didn't like them.

Well, at last, someone up there (Maggie Thatcher, to be precise) has discovered what an amiable fellow our Clive is, and what a splendid computer he has built.

With much pomp, the Industry Minister, Kenneth Baker, announced that in a plan to ensure all 29,000 primary schools in the country had at least one microcomputer, three computers would be officially approved: the BBC microcomputer; the Research Machines 480Z; and the Sinclair Spectrum. The BBC machine and the 480Z were the two approved for secondary schools.

The only fly in this particular ointment is that the Government has seen fit to insist that schools buy one whacking great package (and not just the humber micro itself) to qualify for the pound-for-a-pound assistance offered. When you decide your primary school is going to get a Spectrum under the scheme, as well you have to buy: a monitor, RS232 interface, custom built tray to hold the lot and a cassette recorder. And the Spectrum must be a 48K unit.

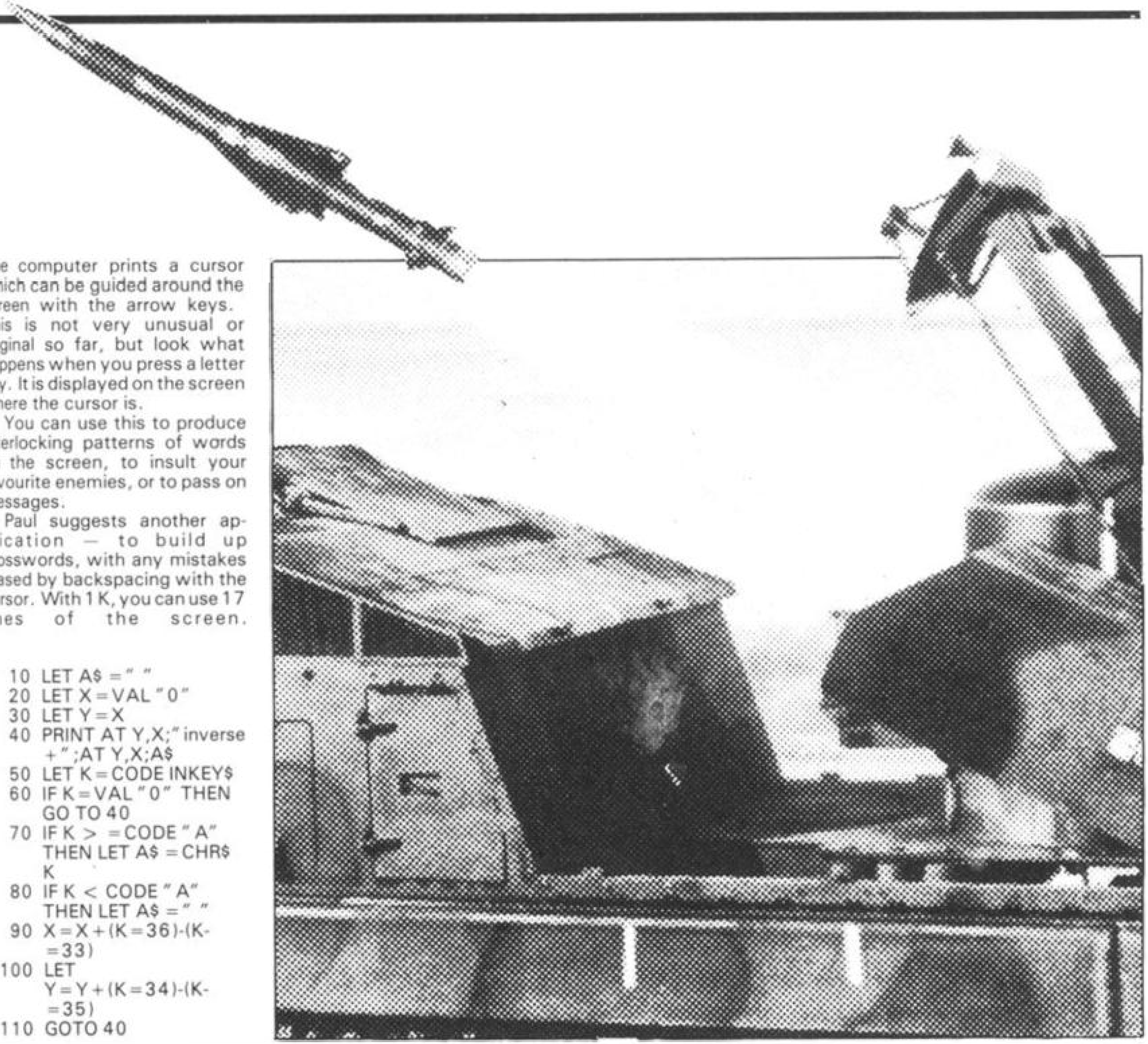
Of course, what a lot of schools are likely to do is, now that the Spectrum has been approved, look at the price of the package (even after getting a little help from their governmental friends), and wisely decide a much better bet would be to buy the Spectrum direct. After all, there is hardly likely to be a school in the country buying a computer which does not have at least one television.



At the press conference when it was announced that the Spectrum was an 'approved micro for primary schools', Kenneth Baker (right, Industry Minister, with Nigel Searle, from Sinclair Research.

Doodle-bug

Here's the perfect way to while away those long summer evenings — with a ZX81 doodle-bug written by Paul Morris of Alford, Lincolnshire.



The computer prints a cursor which can be guided around the screen with the arrow keys. This is not very unusual or original so far, but look what happens when you press a letter key. It is displayed on the screen where the cursor is.

You can use this to produce interlocking patterns of words on the screen, to insult your favourite enemies, or to pass on messages.

Paul suggests another application — to build up crosswords, with any mistakes erased by backspacing with the cursor. With 1 K, you can use 17 lines of the screen.

```
 10 LET A$ = " "
 20 LET X=VAL "0"
 30 LET Y=X
 40 PRINT AT Y,X;" inverse
    + ";AT Y,X;A$
 50 LET K=CODE INKEY$
 60 IF K=VAL "0" THEN
    GO TO 40
 70 IF K > =CODE " A"
    THEN LET A$ =CHR$
    K
 80 IF K < CODE " A"
    THEN LET A$ = " "
 90 X=X+(K=36)-(K-
    =33)
100 LET
    Y=Y+(K=34)-(K-
    =35)
110 GOTO 40
```



ZX 81



Getting things into order

This program is a useful one if you have to sort a list into order, or to produce an index for a book. There are two versions — one for the ZX81 and one for the ZX Spectrum. The programs are by Tim Hartnell.

The programs explain themselves as they run. The first, longer, listing is for the ZX81. It contains an error-checking routine, so that you can amend entries before making them a final part of the list to be sorted and printed.

The Spectrum version does not contain such a routine, but one could easily be added if you so desire, by copying the routine from line 620 in the ZX81 program.





ZX81 LISTING

```
  10 REM INDEX
  20 PRINT "ENTER TITLE"
  30 INPUT T$
  40 PRINT "ENTER AUTHOR"
  50 INPUT A$
  60 CLS
 200 DIM W$(200,32)
 300 LET B=0
 400 LET G=0
 500 LET A=1
 600 PRINT AT 2,0;"ENTER SUBJECT
";A;" AND PAGE",""""E"" TO END"
 610 INPUT W$(A)
 615 IF W$(A)="E
                 " THEN GOTO 950
 620 PRINT "ITEM ";A;" IS"
 625 PRINT
 630 PRINT W$(A)
 635 PRINT
 640 PRINT "IF THIS IS CORRECT,
PRESS ENTER"
 650 PRINT "IF NOT PRESS ANY KEY
 THEN ENTER"
 660 INPUT Z$
 670 CLS
 680 IF Z$<>"" THEN GOTO 600
 685 LET W$(A)=W$(A)( TO 32)
 690 LET A=A+1
 700 LET G=G+1
 710 GOTO 600
 950 CLS
 960 PRINT "DO YOU WANT A PRINTO
UT (1), OR"
 970 PRINT "JUST ON THE SCREEN (
2)?"
 975 INPUT Y
 980 SCROLL
 985 IF Y=1 THEN LPRINT T$,,A$
 990 PRINT T$
 991 SCROLL
 992 PRINT ,A$
 993 IF Y=1 THEN LPRINT
 994 SCROLL
 995 LET Z=1
1000 LET B=Z+1
1010 IF B>G THEN GOTO 1090
1020 IF W$(B)>W$(Z) THEN GOTO 10
50
1030 LET Z=Z+1
1040 GOTO 1000
1050 LET Q$=W$(Z)
1060 LET W$(Z)=W$(B)
1070 LET W$(B)=Q$
1080 GOTO 1030
1090 SCROLL
1100 PRINT W$(G)
1105 IF Y=1 THEN LPRINT W$(G)
1110 LET G=G-1
1120 IF G>0 THEN GOTO 995
```

SPECTRUM PROGRAM

```
  10 REM Book index
  20 DIM A$(500,16)
  30 INPUT "ENTER TITLE ";T$
  40 INPUT "ENTER AUTHOR'S NAME
";N$
  50 FOR G=1 TO 500
  60 INPUT "ENTER WORD AND PAGE
NUMBER          ENTER "E" TO END ENT
RIES ";A$(G)
  70 REM 15 SPACES IN NEXT LINE
  80 IF A$(G)="E               "
 THEN GO TO 200
  90 PRINT A$(G)
 100 NEXT G
 200 CLS
 210 PRINT "STAND BY, SORTING"
 220 FOR B=1 TO G-1
 230 FOR C=B+1 TO G-1
 240 IF A$(B)<=A$(C) THEN GO TO
280
 250 LET D$=A$(B)
 260 LET A$(B)=A$(C)
 270 LET A$(C)=D$
 280 NEXT C
 290 NEXT B
 300 PRINT '"READY"
 310 PRINT "ENTER 1 TO LPRINT LI
ST"
 320 PRINT '"ENTER 2 TO PRINT ON
SCREEN"
 330 IF INKEY$="2" THEN GO TO 44
0
 340 IF INKEY$="1" THEN GO TO 36
0
 350 GO TO 330
 360 LPRINT T$
 370 LPRINT
 380 LPRINT N$
 390 LPRINT
 400 FOR A=1 TO G-1
 410 LPRINT A$(G)
 420 NEXT A
 430 STOP
 440 PRINT 'T$
 450 PRINT 'N$
 460 PRINT
 500 FOR A=1 TO G-1
 510 POKE 23692,-1
 520 PRINT A$(G)
 530 NEXT A
```

```
THE OUTSIDER
                  WILSON, COLIN

DOSTOEVSKY - 179
LAWRENCE - 28
LAWRENCE - 87
STEPPENWOLF - 72
ZARATHUSTRA - 150


USING LONDON
                  ROYDS AND TUFT

ACCOMMODATION - 17
BETHNALL GREEN MUSEUM - 95
CITIZENS ADVICE BUREAU - 163
DANCING - 23
GUILDHALL LIBRARY - 99
HAMPSTEAD - 119
HISTORY STUDIES - 117
LIFE SAVING - 89
OPEN UNIVERSITY - 134
RED CROSS - 51
SUPPLEMENTARY BENEFITS - 146
TOWER THEATRE - 35
UNITARIAN CHURCHES - 76
WOMENS LIBERATION WORKSHOP - 7
YOUNG THEATRE - 37
```

Zap! Pow! Boom!

The Psion software company got the big deal that everyone wanted — to become the 'official' Sinclair software supplier. How good is their material? Did they deserve the prize? Nick Pearce takes a look.

Avid readers of the computer press, and visitors to the ZX81 counters of W H Smith, could hardly have failed to notice the extensive range of software recently released by Sinclair. I decided to look at six of the Psion cassettes.

The best of the six is, in my opinion, FLIGHT SIMULATION. This is a superb program which makes very good use of the power and the graphics of the ZX81.

You are the pilot of a small, high-performance aeroplane, which must be safely landed. You have a choice of three screen displays. The *cockpit display* shows the outside world (horizon) in the upper half of the screen, and cockpit instruments in the lower half, including power indicator, altimeter, fuel guage, rate of climb indicator, air speed indicator and radio direction finding equipment.

The *map* shows the position of the runway, various radio beacons, the aeroplane, and an escarpment of hills to hinder your landing.

The third display is the *visual approach* which shows a full perspective view of the runway lights, together with some essential landing indicators.

This display is particularly effective and gives a convincing "pilot eye" view of the runway lights which cope admirably with banking and changes in speed or altitude.

The program's instructions are clear and concise. It is well worth getting acquainted with them before flying. I didn't, and consequently landed with the undercarriage up, and then made repeated attempts to land without using flaps, and therefore stalling, on my next few flights.

After a little practice, flyiing becomes easier, and full use can be made of the controls and facilities that were ignored during the panic of the first few flights.

An option at the start of the program allows the aspiring aviator to practice and perfect just the final runway approach and landing. If you find flying becomes too easy — which is most unlikely — you can introduce additional complexity by adding wind effects.

I have just two criticisms of FLIGHT SIMULATION. I would have liked to try my hand at taking-off, which is not possible with this program; the program permits the pilot to 'cheat' and 'fly' at ground level (zero altimeter reading) on the runway approach. Apart from these points, I could not fault this software. At £5.95 it represents very good value for money.

Incidentally, a friend kindly lent me a 12½K flight simulation on his Tandy computer. In many respects I found the Psion simulation superior; both its visual effects and navigational facilities were more convincing, and it was more user-friendly. The Tandy version does have some additional features, however, including take-off and taxiing, and a wargame facility.

SPACE RAIDERS is a ZX81 version of the 'traditional' arcade game SPACE INVADERS in which you defend the earth from successive armies of invaders.

You have three lives (bases), and points are awarded for each raider ship hit. RAIDERS may be played at one of three speeds, normal, fast and super-fast, and I was told by an experienced player who had a go that it bears the most similarity to the arcade game when it is played in the superfast mode, which was too fast for me.

This program is similar to the many other invaders programs on the market, but has the bonus of a second game, BOMBER, on the B side.

In this 'blitz'-type game, you use bombs and rockets in an attempt to raze a skyscraper city to the ground before your plane either crashes, or is knocked out by the blast from one of your own rockets. It is probably impossible to completely obliterate the city. I tried in vain several times at the slowest speed (and there are nine speeds altogether). Not a particularly inspired game, I thought, but one which makes a welcome change from invaders.



A nice finishing touch to RAIDERS and BOMBER is the display at the end of each game which gives the score for that game and the highest score so far. In common with the rest of the software in this review, these programs are pretty well idiot proof, and I (even I) could not get either of them to crash.

Both these games perform well, and at £3.95 for the two, this cassette is a reasonable buy.

Moving on to programs for

the more serious user, the Sinclair range includes VU FILE and VU CALC; programs intended for business or household management use.

VU FILE is a general purpose filing program. It is the sort of program you could use to catalogue *your coin or stamp* collection, keep a name and address list of friends, or for business, maintain a membership record for your club or society, or even keep a file record of all your ZX81 software!

The program is logically displayed, and easy to use. You start by 'laying out' the record, using headings, titles and graphics symbols. Anything entered during this *record layout* mode will be shown on each record on the file. You then move to another file, *datafields,* in which you define the positions on the record where you wish to set the information.

Data is entered into the file from the *main command* mode. As well as ENTER, the other main commands are ALTER, INFORM, FORWARD, BACK, RESET, ORDER, PRINT, COPY, SELECT, QUIT, LIST and DELETE. These provide a comprehensive file handling capability and can be implemented simply by keying in just the first letter of each command.

The commands are pretty much self-explanatory, but where there is any difficulty in comprehension, the instructions give a fuller explanation.

The top of the screen gives continuous prompts to guide the user through the system.

Recorded on the B side of this cassette is an example of an application of VU FILE. It is called GAZETTEER, and is a file of records for every country in the world, giving the name of the country, its capital city, main languages and the like.

I made use of GAZETTEER to get a feel for the manipulative and interrogative power of the main program.

I was able to select and view any country in the file almost instantaneously; in well under a second. The record to be selected must be correctly defined. For example, the USA is filed as UNITED STATES OF AMERICA, and Russia as USSR. On the command LIST, each record in the file is displayed for about one and a half seconds.

By using the commands FORWARD and BACKWARD you can step through the file. RESET sends you to the first record, ORDER defines the particular sequence required, ordering being defined by the leading alphanumeric characters of any of the data fields.

INFORM gives information on the status of the file. As an indication of the capacity of VU FILE, the GAZETTEER holds records on 152 countries, and uses 96% of the space available in the program.

I think this software might be let down, not by the quality of the program itself (which I thought very high indeed), but by the inconvenience of loading the files from cassette. Few people would be prepared to wait for five minutes or more to load a file to obtain the address of a friend, or colleague. The ZX81 really needs a disc operating system to be used effectively for this sort of work.

For business, or perhaps school or hobby use where a cassette can be loaded at the beginning of the day or session, and the ZX81 can remain dedicated for some time, VU FILE on a cassette might be viable, although file space may quickly become a limitation.

Although the GAZETTEER effectively demonstrates the power and capacity of VU FILE, and while it clearly has initial novelty value, I think I would prefer to browse through a good atlas to get the kind of information the program contains.

VU FILE is, however, a good program, and does all that is claimed for it. The real question, however, is whether it would actually ever be used for all those filing jobs you've always meant to do.

The program certainly gave me a better understanding of how computers can be used for organising data, and the power a computer system can offer.

The discipline of attempting to catalogue on VU FILE that coin or stamp collection can itself be a worthwhile exercise, whether or not the computer eventually replaces your manual records.

VU CALC — a ZX81 version of VISICALC — is a program for calculating and displaying tables of numbers and names. You start with an empty table of 'boxes' in a grid of 26 rows by 36 columns. Only a small part of the table can be displayed at one time, but you can quickly move around the *table using the arrow keys.*

Using simple commands, you can enter data into the boxes, and use formulae to link boxes, rows or columns, so that the computer can quickly calculate a complete table.

The power of the program lies in the ease and speed with which the table can be recalculated with altered data or different formulae.

There is a facility for saving the program on cassette, together with the completed or partially completed table, and it could easily be used for something like monthly accounting.

As an engineer, I was rather disappointed that this program only permits the four basic arithmetic operations (+ − / *) to be carried out. Many engineering applications would require logarithmic and/or trigonometric functions as well. For this reason, I think VU CALC is more likely to be used as a tool for financial analysis, rather than engineering, scientific or statistical applications.

I had some difficulty assimilating the operating techniques required by this program, and I had to persevere to make it work. However, after rereading the instructions and some practice, I found I was able to operate the program quite efficiently.

Once the formulae and titles for a particular chart had been developed and entered, it is quite easy to enter the data into the chart from which the complete table can be calculated.

Some care needs to be taken in setting up the formulae, and entering data. For example, you must ensure that when operations are carried out, the resulting figures can be displayed without problems in the eight digit boxes.

VU CALC could be a useful tool for managing household accounts, although I am not sure it could compete with some of the dedicated financial control programs on the market which are designed for more sophisticated applications.

A lot of effort has gone into producing VU CALC, and it works well, again with considerable emphasis on user-friendly aspects. However, I feel its field of application is likely to be limited since it rather lies between two camps. It is neither a proper financial analysis program, nor is it ideally suited for serious use by the engineer or statistician.

For some household applications, VU CALC (like VU FILE), could prove a boom. You do need a certain amount of dedication to set up and maintain a computerised system, but the result is likely to be well worth the effort.

Had any good fantasies lately?

Finally, on a lighter side, is FANTASY GAMES, which contains two 'adventure' programs.

On the A side of the cassette is PERILOUS SWAMP, which Psion recommend you tackle before you take on the more difficult SORCERER'S ISLAND on the B side.

When you enter the Perilous Swamp you must rescue a princess who has been captured by an evil wizard, and then return safely.

The game starts with a map showing your position, along with that of the princess, and the locations of several swamps in the rescue area.

The map, which can be called up at any stage, is different for each game. As you move, prompted by requests for compass directions, mosters and treasures are encountered, and you gain or lose points as you fight or bribe your way onwards.

This program has been well written and is quite entertaining. However, it is more of a guessing game than a true adventure program.

SORCERER'S ISLAND is a cross between a guessing game like Swamp, and more traditional adventures. You have been marooned on the island from which you must escape. A detailed map can be called up at will, although the screen goes blank for nearly a minute each time you do so.

You have a vocabulary of some ten words with which you tell the computer what you want to do. At each turn, your immediate surroundings are shown. Again, these take rather a long time to display.

As the game progresses, you are assailed by monsters, tempted by treasures, and have to deal with the other hazards that abound on the island.

There is supposed to be a grand sorcerer who can help you escape, but I have to admit I never managed to meet him.

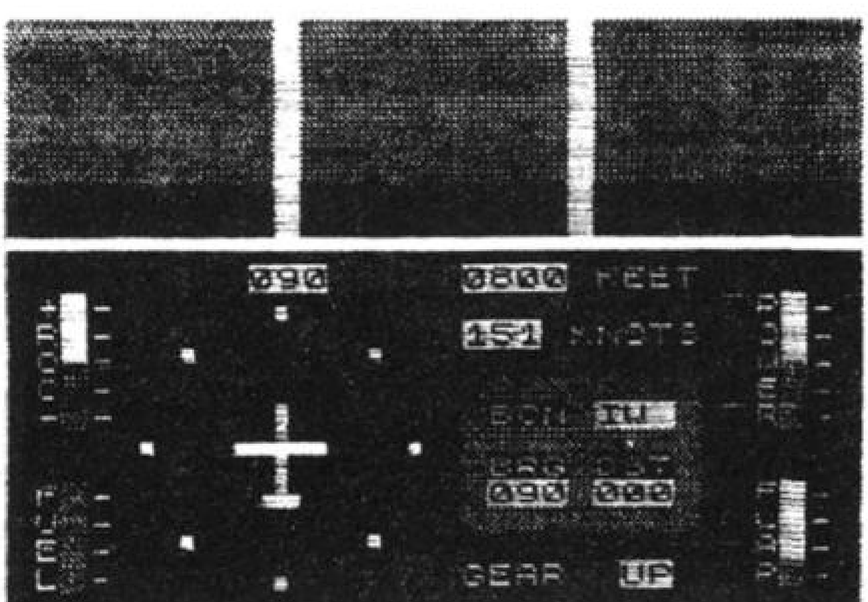
Again, an entertaining game, although I did think it was spoilt to some extent by the length of time it took to create the screen displays. FANTASY GAMES costs £4.75.



F=FORMULA D=DATA C=CALCULATE

	01	02	03
A	BILLS	JANUARY	FEBRUARY
B	MORTGAGE	155.75	155.75
C	RATES (G)	0.0	0.0
D	RATES (W)	0.0	0.0
E	ELECT.	58.11	0.0
F	TELE.	0.0	40.12
G	FUEL	45.00	45.00
H			
I	TOTAL	258.86	240.87

Printout from VU CALC



Cockpit display from FLIGHT SIMULATION

GAZETTEER
COUNTRY
PERU
CONTINENT S.AMERICA
CAPITAL LIMA
LANGUAGES QUECHUA,SPANISH
CURRENCY SOL
POPULATION 16.0 MILLION
GNP PER CAPITA 710 US$ (1979)
LAND AREA 1285,000 SQ.KM.
STATISTICS OF THE WORLDS NATIONS

VU-FILE Gazetteer output



The cassettes

Summary

Of the six cassettes I reviewed, I was particularly impressed by FLIGHT SIMULATION. It is a novel and apparently authentic simulation which must have taken considerable effort to compile, and of which Psion should feel proud.

All the programs maintained a high standard, and together must represent some of the better software available for the ZX81.

The price of each is more than reasonable, with FLIGHT SIMULATION being particularly good value.

Each cassette is attractively boxed, and well presented, with clear and concise instructions on the cassette insert card. All the programs loaded first time, an indication of the excellent quality of the recordings.

I was interested to see that CRC (cyclic redundancy check) to which each program is subjected on loading. This checks that the program has not corrupted and should give the user some confidence that the program will not crash. It is also used by the manufacturer as a check on recording quality.

All the cassettes are for the ZX81 and can be obtained from W H Smith branches, or from Sinclair Research, Freepost, Camberley, Surrey, GU15 3BR.

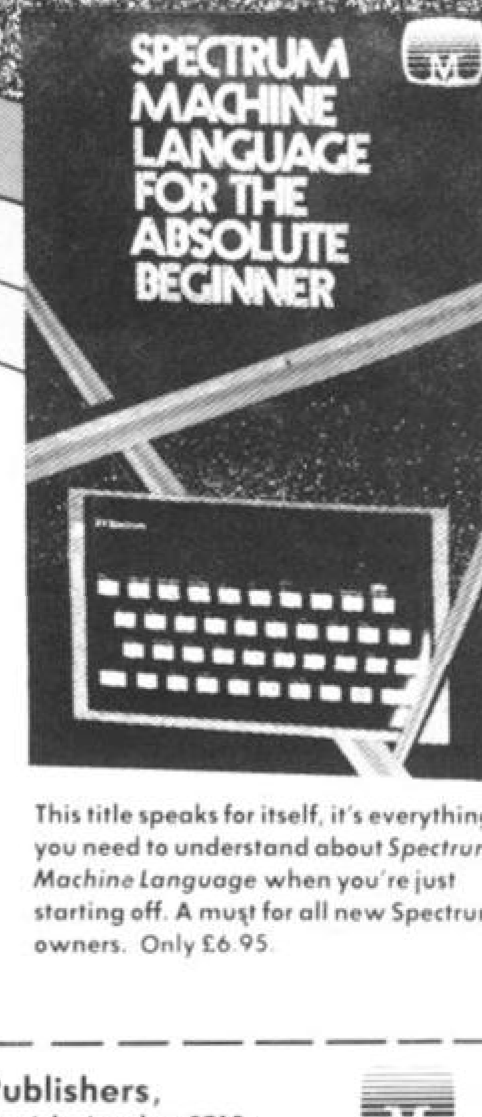
ANNOUNCING The BEST Books For Your SPECTRUM



Dr. Ian Logan is the acknowledged leading authority on Sinclair computers. In this book, he gives a complete overview of the way the Spectrum operates, both for BASIC and machine language programming. A special section on the ROM operating system will give you insight into this computer as well as provide you with information on how to use many of the routines present in the ROM. This book is a must if you are serious about programming the Spectrum. Only £7.95.



Over the Spectrum is the book where your dreams really do come true! With the full listing of over 30 programs for your Spectrum, this book will show you how to use the computer's complete facilities. Exciting games such as *Meteor Storm, Eliminator* and *Spectrum Invaders*, together with utilities, educational programs, programming tips and hints, make this the definitive book for every Spectrum user. Only £6.95



This title speaks for itself, it's everything you need to understand about *Spectrum Machine Language* when you're just starting off. A must for all new Spectrum owners. Only £6.95.

After leading the way in Sinclair ZX81 software, we've produced the highest quality, most exciting Spectrum software available. From the three excellent books depicted above to fast-action games on cassette, we're providing the best choice in Sinclair Spectrum software today.

Whether it's for your new Spectrum or ZX81 Melbourne House has books and programs perfectly suited to your needs.

Send for your Spectrum or ZX81 catalogue today.

Melbourne House Publishers,
131 Trafalgar Road, Greenwich, London SE10.
Correspondence to: Glebe Cottage, Station Road, Cheddington, Leighton Buzzard, BEDS LU7 7NA.

☐ Please send me your Spectrum/ZX81 catalogue (please specify).
☐ Please send me .... (please add 80p for post, pack & V.A.T.)
Name ....
Address ....
.... Post Code ZX10

MELBOURNE HOUSE PUBLISHERS

Navarone ablaze!



A few user-defined graphics here, a blob of colour there, a BEEP or two to keep the neighbours bemused, and Ken North of Ashford, creates this program to keep trigger fingers twitching



This program shows an aircraft flying from right to left above an anti-aircraft battery. The A.A. guns are firing at the aircraft, and the aircraft can drop bombs to destroy the guns. You drop a bomb by pressing the 'M' key.

The A.A. fire reduces when you knock out a gun, but beware: the gunners are very accurate and have found your exact height, so you must knock a gun out very quickly to survive a round.

The main program runs from line 10 to 299, and the REMs explain what the other routines do.

The first section, lines 10 to 100, sets up the screen. Line 10 calls the routine to set up the user-defined graphics. The data for these were converted to decimal numbers, and then stored in lines 1100 to 1120. These are then read and POKED into the appropriate character. The aircraft uses the graphic characters "A", "B" and "C". The bomb is "D" and the gun is "E".

Lines 20 and 30 set the screen colours to a white sky, green ground and a black border.

The CLS command, after changing PAPER colour is, as you probably know, important. If you don't CLS, the old paper colour remains, and only changes when you print something, which can give you some strange and colourful (although unwanted) effects.

Lines 40 and 60 set up the variables.

Lines 70 to 90 find the gun positions, line 85 checks for three different positions, and line 100 prints the guns.

The "E" is a graphics "E" which will change to the gun after the first call to subroutine 1000.

Lines 110 to 200 are the loop for the aircraft printed in line 120. The "A", "B" and "C" are the graphics and, in line 100, the trailing space is to clear the last character as the aircraft moves across the screen. Line 140 checks for key "M" and sets a flag F1. If set, the program jumps to the 'bomb drop routine'.

If clear, a PAUSE is put in so that the program runs at the same speed as it does when a bomb is dropped.

The A.A. fire is controlled by line 170. The values can be changed to make the game easier or harder, but the 1.8 seemed a good compromise. Line 180 again puts in a PAUSE to keep the routine the same, whether the guns fire or not.

The variable HITS in line 190 checks to see if all three guns have been destroyed, and if it finds that they have, increments the score, and then branches to clear the ground and print a new battery. The rest of the main program prints your score after you have lost all your aircraft.

The BOMB DROP routine prints the bomb if F1 is set. If the bomb hits the ground, F1 is reset for the next keypush.

The check, to see if you have hit a gun, uses the ATTR function in line 340. The ATTRibute call returns a value — in binary — which depends on certain conditions:
The MSB (bit 7) is for FLASH, bit 6 is for BRIGHT, bits 5 to 3 for PAPER colour and bits 2 to 0 for INK colour.

In case of the gun, it is not flashing, and is normal brightness, so bits 7 and 6 are at zero. The PAPER is white, so bits 5 to 3 are all ones and the INK is red, so bit 1 is 1 and bits 2 and 0 are zero.

This gives a BIN00111010 which, when converted to decimal, gives 58. If the program returns 58 at line 340, it branches to 400, the SCORE and BANG routine. This causes the guns to explode . . . and increments the score.

The A.A. routine called by line 170 puts a flash on the screen, and checks if it coincides with the nose of the aircraft. If it does, it calls the SHOOTDOWN routine at line 600.

SHOOTDOWN makes the aircraft crash, flashing as it does so. The flash is confined to the plane, as it is locally defined in the PRINT statement, and therefore ends with the statement.

A PAUSE follows, to allow you to get your breath back, before the action starts again. You get six aircraft to try and destroy as many guns as you can.

If you want to make changes in colour, or to the A.A. fire, the only line you have to be careful of is 340 with the ATTR function call.

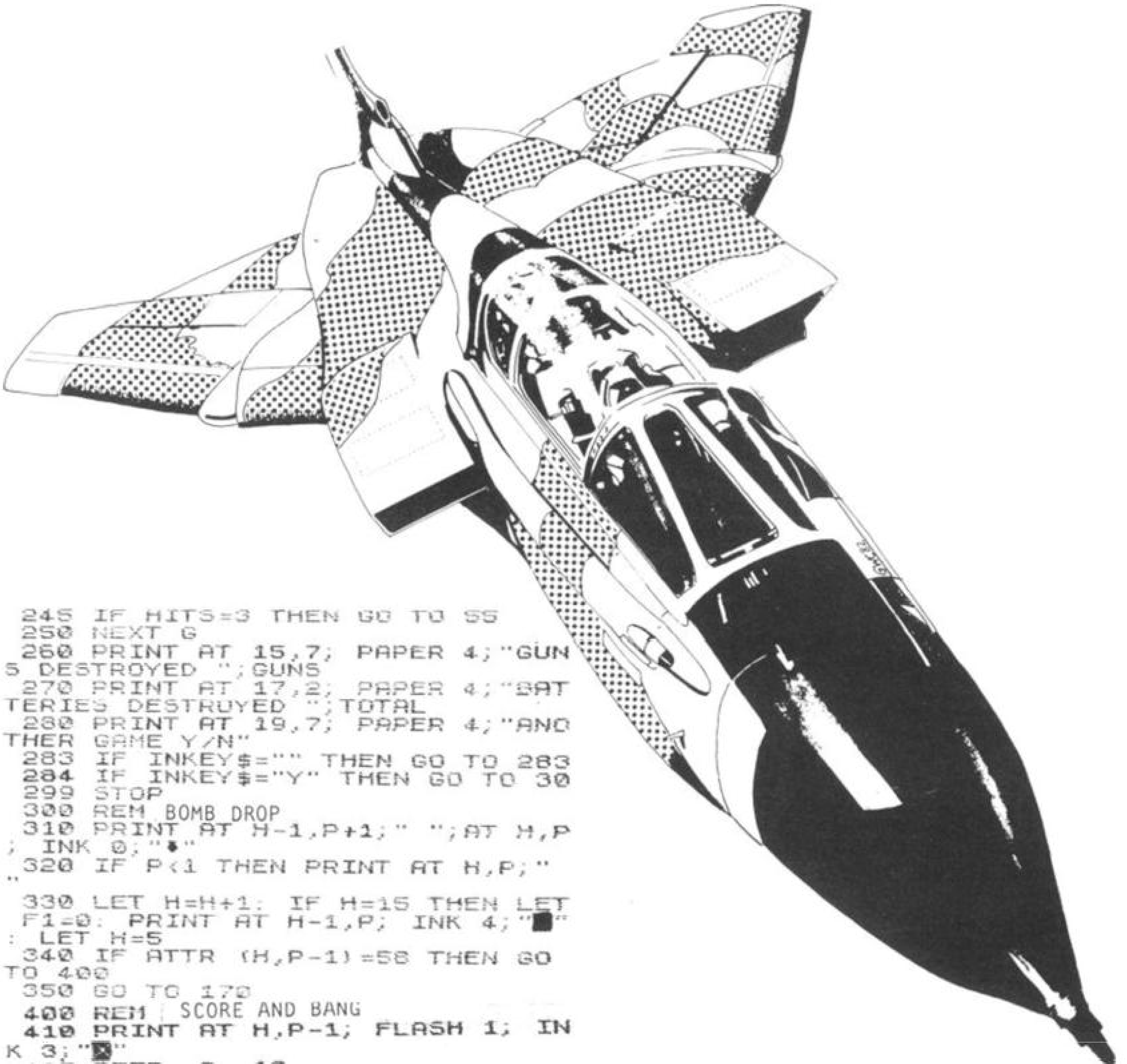
If you change the sky colour (PAPER), or the gun colour (INK), you will have to work out the corresponding ATTRibute, referring to page 116 of the manual.

Another thing to watch is whether the keyboard is in the C or L mode for the bombdrop. I've made the program run for the C mode, but line 140 could be changed to read both by changing it to: 140 IF INKEY$ = "M" OR INKEY$ = "m" THEN LET F1 = 1.

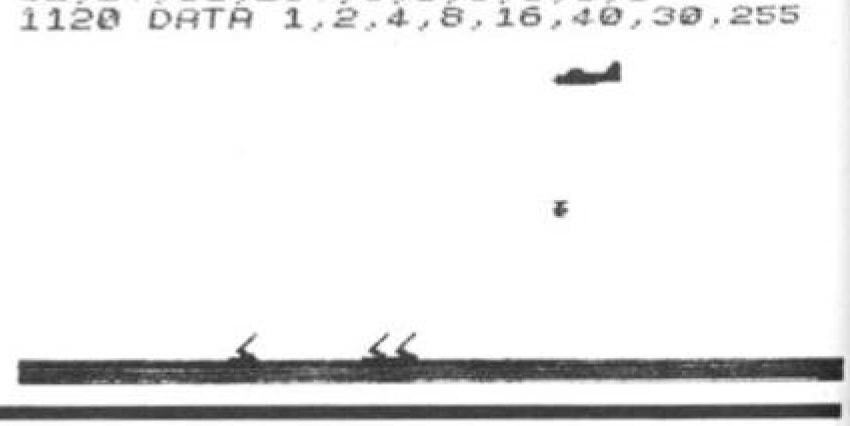


PROGRAM LISTING

```
  10 GO SUB 1000
  20 BORDER 0: PAPER 7: CLS
  30 FOR L=14 TO 21: FOR J=0 TO
31: PRINT AT L,J; INK 4;"█": NEX
T J: NEXT L
  40 LET TOTAL=0: LET GUNS=0: LE
T T=6
  50 FOR G=0 TO 5
  55 PRINT AT 15,7; PAPER 4; INK
 0;"PLANES LEFT ";T-G
  60 DIM Q(3): LET H=5: LET F1=0
: LET HITS=0
  70 FOR L=1 TO 3
  80 LET W=INT (RND*14)+5: FOR N
=1 TO 3
  85 IF W=Q(N) THEN GO TO 80
  87 NEXT N
  90 LET Q(L)=W: NEXT L
 100 FOR L=1 TO 3: PRINT AT 13,Q
(L); INK 2;"": NEXT L
 110 FOR P=28 TO 0 STEP -1
 120 PRINT AT 3,P; INK 0;"    "
 130 BEEP .005,-10: BEEP .005,-4
 140 IF INKEY$="M" THEN LET F1=1
 150 IF F1=0 THEN PAUSE 3
 160 IF F1=1 THEN GO TO 300
 170 IF RND*(5-HITS)>1.8 THEN GO
 TO 500
 180 PAUSE 2
 190 IF HITS=3 THEN LET TOTAL=TO
TAL+1: GO TO 240
 200 NEXT P
 210 PRINT AT 3,0;"     "
 220 GO TO 110
 240 FOR L=0 TO 31: PRINT AT 12,
L;" ";AT 13,L;" ": NEXT L: PRINT
 AT 3,P;"     "
```



```
 245 IF HITS=3 THEN GO TO 55
 250 NEXT G
 260 PRINT AT 15,7; PAPER 4;"GUN
S DESTROYED ";GUNS
 270 PRINT AT 17,2; PAPER 4;"BAT
TERIES DESTRUYED ";TOTAL
 280 PRINT AT 19,7; PAPER 4;"ANO
THER GAME Y/N"
 283 IF INKEY$="" THEN GO TO 283
 284 IF INKEY$="Y" THEN GO TO 30
 299 STOP
 300 REM BOMB DROP
 310 PRINT AT H-1,P+1;" ";AT H,P
; INK 0;"▼"
 320 IF P<1 THEN PRINT AT H,P;"
"
 330 LET H=H+1: IF H=15 THEN LET
 F1=0: PRINT AT H-1,P; INK 4;"■"
: LET H=5
 340 IF ATTR (H,P-1)=58 THEN GO
TO 400
 350 GO TO 170
 400 REM SCORE AND BANG
 410 PRINT AT H,P-1; FLASH 1; IN
K 3;"■"
 415 BEEP .2,-10
 420 LET HITS=HITS+1: LET GUNS=G
UNS+1
 450 GO TO 170
 500 REM ACK/ACK
 510 LET R=INT (RND*25)
 520 PRINT AT 3,R; INK 0; FLASH
1;"#"
 530 IF R=P THEN GO TO 600
 540 PRINT AT 3,R;" "
 550 FLASH 0: GO TO 200
 600 REM SHOOTDOWN
 610 FOR L=3 TO 13
 620 PRINT AT L,P; FLASH 1; INK
2;"▬"
 630 PRINT AT L-1,P;"    "
 640 BEEP .1,L+5
 650 NEXT L
 670 PAUSE 500: LET F1=0
 680 FOR L=0 TO 31: PRINT AT 13,
L;" ": NEXT L
 690 GO TO 250
 999 STOP
 1000 REM GRAPHICS SETUP
 1010 FOR A=0 TO 7
 1020 READ D: POKE USR "A"+A,D: R
EAD D: POKE USR "B"+A,D
1030 READ D: POKE USR "C"+A,D. R
EAD D: POKE USR "D"+A,D
1040 NEXT A
1050 FOR A=0 TO 7: READ D: POKE
USR "E"+A,D: NEXT A
1060 RETURN
1100 DATA 0,0,12,0,0,0,28,60,1,2
24,60,24,3,255,252,60
1110 DATA 31,255,252,60,63,255,2
52,24,31,254,0,0,0,0,0,0
1120 DATA 1,2,4,8,16,40,30,255
```



MAIL ORDER PROTECTION SCHEME

If you order goods from Mail Order Advertisers in this magazine and pay by post in advance of delivery, this publication will consider you for compensation if the advertiser should become insolvent or bankrupt, provided:

1. You have not received the goods or had your money returned; and
2. You write to the publisher of this publication explaining the position not earlier than 28 days from the day you sent your order and not later than 2 months from that day.

Please do not wait until the last moment to inform us. When you write, we will tell you how to make your claim and what evidence of payment is required.

We guarantee to meet claims from readers made in accordance with the above procedure as soon as possible after the advertiser has been declared bankrupt or insolvent to a limit of £1,800 per annum for any one advertiser, so affected, and up to £5,400 p.a. in respect of all insolvent advertisers. Claims may be paid for higher amounts, or when the above procedures have not been complied with, at the discretion of this publication, but we do not guarantee to do so in view of the need to set some limit to this commitment and to learn quickly of reader's difficulties.

This guarantee covers only advance payment sent in direct response to an advertisement in this magazine (not, for example, payments made in response to catalogues, etc, received as a result of answering such advertisements):

CLASSIFIED ADVERTISEMENTS ARE EXCLUDED.

Structured programming

Although the art of programming is not a discipline with a long history, a consensus has grown up regarding the merits of writing programs in a structured way. In this article, Tim Hartnell briefly introduces some of the key ideas of structured programming.

You've probably gone through several stages as you develop your programming skills. After the first, brief struggle with BASIC, you suddenly discovered you could, after a fashion, write programs which ran. They may have looked pretty convoluted when you looked at their listings, and friends may have needed a detailed explanation from you before they knew what to do when running the programs, but at least they worked.

There comes a stage when you decide you're going to have to do better than that. But while you may be vaguely dissatisfied with your programs, you may not have much idea of how to go about becoming a better programmer. Here are a few guidelines which may help.

First, have a look at a printout of your listing. Programs linked by REM statements look better, and are easier to understand when you return to them after a break. Of course, shortage of memory may preclude the luxury of REM statements, but if you have the memory, you should include them. REM statements filled just with a line of asterisks can prove quite useful in separating each major section of the program. Examine any unconditional GOTO critically. Too many GOTOs leapfrogging over other parts of the program show a lack of directed thinking, make programs run more slowly, and can make them almost impossible to decipher.

It is very good programming practice, though not the most memory—efficient way to go about writing a program, to have each of the main sections of the program (like the one which assigns the variables at the beginning of a run, the one which prints out the board, the one which works out who has won, and so on) in separate subroutines. The beginning of a board game program could well look like this:

```
10    REM *NAME OF PRO-
      GRAM*
20    REM ASSIGN VARIABLES
30    GOSUB 9000
40    REM PRINT BOARD
50    GOSUB 8000
60    REM HUMAN'S MOVE
70    GOSUB 7000
80    REM COMPUTER'S MOVE
90    GOSUB 6000
100   REM CHECK IF GAME
      OVER
110   GOSUB 5000
120   GOTO 50
```

As you can see, this ensures that the program actually cycles through a continuous loop over and over again, until the program terminates within the "CHECK IF GAME OVER" subroutine. You can actually write a series of lines like these before you start writing anything else, and even before you know how you are going to actually perform some of the tasks within the subroutine.

Then you can write the program module by module, making sure that each module works before going onto the next. It is relatively easy to debug a program like this, and far simpler to keep an image of 'where everything is' when you do this, than when you just allow a program to, more or less, write itself.

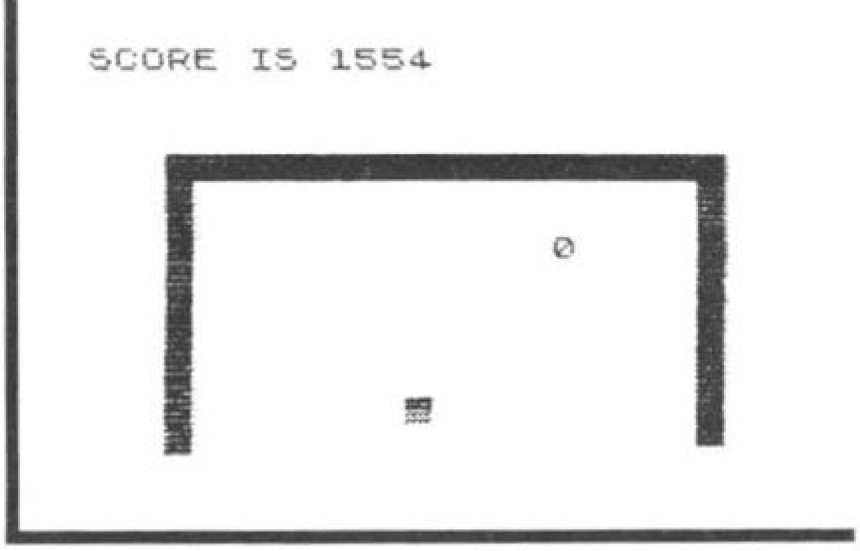
The listing should be, then, as transparent as you can make it, both for your own present debugging, and for future understanding of what bit carries out what task. The output of the program should also look good. Again, if memory is not a problem, make sure the display is clear and uncluttered. Use blank PRINT lines to space it out, use rules of graphic symbols or whatever to break the screen up into logical sections and so on. Once you have a program working satisfactorily, it is worth spending extra time on the subroutine which controls the display. Here you'll appreciate again the advantage of having all the display handling in one subroutine, as it will be easy to know where to go to enhance the display.

Of course, as we live in a far from ideal world, it is unlikely that every single display command can be contained within one subroutine, but if you aim towards that end, it will make subsequent working upon the program much easier than it might be otherwise.

You can make the program even easier to read by assigning explicit variable names to the numbers which refer to the subroutines. By this I mean, in the case of the example given a little earlier, that you assign the value of 8000 to a variable called PRINTBOARD and 7000 to a variable called HUMANSMOVE. Then, the main loop will contain lines like GOSUB PRINTBOARD and GOSUB HUMANSMOVE so you know exactly what the subroutine call will generate. You'll see this in the program SQUASH which comes at the end of this article in which the things the program does include moving the ball (LET MOVEBALL = 550), assigning the variables at the beginning of the game (SETUP), and moving the bat (MOVEBAT). This makes programs very easy to follow.

The 'structured' approach outlined also helps you realise another aim of a good program — to do what you expected it to, every time you run it. You should write a program so that, even if

```
  10 REM SQUASH
  15 REM AFTER PROGRAM BY
         JEREMY RUSTON
  20 REM MOVE BAT WITH Z AND M
         KEYS
  25 LET MOVEBALL=550
  30 LET SETUP=300
  35 LET MOVEBAT=460
  40 LET HIGHSCORE=0
  50 LET SCORE=0
  60 GOSUB SETUP
  70 REM ************
  80 LET SCORE=SCORE+INCREMENT
 110 LET A$=INKEY$
 130 IF A$="Z" OR A$="M" THEN GO
SUB MOVEBAT
 140 GOSUB MOVEBALL
 150 PRINT AT 19,B+11;B$
 160 GOTO 80
 290 REM ***********
 300 REM ** SET UP **
 310 LET X=1
 320 PRINT AT 10,10;"█████████████
█████████"
 330 FOR T=0 TO 10
 340 PRINT AT T+10,10;"█";AT T+1
0,30;"█"
 350 NEXT T
 360 LET B$=" █ "
 380 LET Y=1
 385 LET L=1
 390 LET M=1
 400 LET B=10
 410 PRINT AT 19,11+B;B$
 420 LET INCREMENT=207+INT (RND*
100)
 430 RETURN
 450 REM **********
 460 REM ** MOVE BAT **
 480 IF A$="M" AND B=16 THEN RET
URN
 490 IF A$="Z" AND B=0 THEN RETU
RN
 510 IF A$="M" THEN LET B=B+1
 520 IF A$="Z" THEN LET B=B-1
 530 RETURN
 540 REM **********
 550 REM ** MOVE BALL **
 570 PRINT AT 11+Y,11+X;" "
 580 IF L+X>18 OR L+X<0 THEN LET
L=-L
 590 IF M+Y>8 OR M+Y<0 THEN LET
M=-M
 600 LET X=X+L
 610 LET Y=Y+M
 620 PRINT AT 11+Y,11+X;0
 622 IF Y<>8 THEN RETURN
 625 PRINT AT 6,7;"SCORE IS ";SC
ORE
 630 IF Y=8 AND ABS (B-X)<=2 THE
N RETURN
 640 PRINT "END OF GAME"
 660 PAUSE 4E4
 670 CLS
 680 RUN
```



you are not present when a friend decides to run it for the first time, it performs as expected. This means not only, of course, that it is properly debugged, but that the instructions (which can be contained within the ASSIGN VARIABLES subroutine) are clear and complete.

The user prompts should be clear, so the human operator knows whether to enter a number, a series of numbers, a word, a date, a mixture of letters and numbers, and so on. The program has to assume that the operator is a complete idiot, and that no matter how clearly the instructions and/or user prompts are stated, he or she will attempt to do things the wrong way. A classic example of this is the entering of dates. 'Mug traps', as the routines to reject erroneous input from the operator are called, should be set up to reject a date being entered in a form which the computer cannot understand (such as the month before the day) or which is clearly wrong (such as entering the 32nd of February). You should ensure that, no matter what the operator does, the program does not crash or otherwise misbehave. This can happen if the program was expecting a numerical input, and the operator tried to enter a letter or a word, or hit ENTER RETURN without entering anything at all. You can get around this by always allowing a string input, going back for another input if the empty string is entered, and taking the ASC, VAL or CODE of the input to turn it into numerical form.

Documentation is an area of programming which is often neglected. It is virtually essential for a program which is intended for publication, and most advisable for long programs which you've written for yourself. At the least, the documentation should include a list of variables, an explanation of the program structure (which should be easy to do if you've followed the 'modular' approach advised earlier), and brief instructions, especially if the program itself does not contain instructions. A sample run showing the kind of inputs, and the nature and layout of the program outputs, is also useful.

Your program should run as quickly as possible. Every time there is a subroutine or GOTO call, the computer must search through the whole program, line by line, to find the specified line number, so placing often used subroutines near the beginning of the program will speed them up fractionally. That is why the instructions are often placed right at the end. You do not want the computer to have to wade through the initialisation and instruction lines every time it has been told to GOTO or GOSUB looking for the destination, or return line number.

Define often used variables first, so they will occupy the early slots in the variables store. The computer will search the store only until it finds the variable it wants, so there is no point in getting it to look at more entries than absolutely necessary.

Finally, and this is by far the best way to test a program you've written, call in a friend and sit him or her in front of the TV, and tell them to press RUN, without you saying anything, and just sit back and watch. If there is any hesitation, or the program hiccups, you have more work to do.

In summary, then:

* Use REM statements
* Make program listing neat and logical
* Use structured programming techniques, controlling the program through a loop of subroutine calls
* Examine unconditional GOTO commands critically
* Make output display attractive and clear
* Ensure all user prompts are clear
* Add 'mugtraps' on all user input
* Document your programs, even if you just make a list of variables
* Make your program run as quickly as possible
* Test programs by allowing someone unfamiliar with the program to run it

Keeping things in proportion

Tim Rogers of Richmond turns his programming skill to solving the problem of messy word output.

This 'proportional spacing' program not only ensures that words are not split on the end of lines, but also 'pads out' each line in order to use all 32 characters across.

You just type in each word, pressing NEWLINE/ENTER instead of the space key between the words, and the text will be sent to the printer from time to time.

You can correct mistakes within individual words by using the 'less than' sign. When you've finished, press the 'greater than' sign, and all the remaining words will be printed.

This program should give a more 'professional' look to your text.

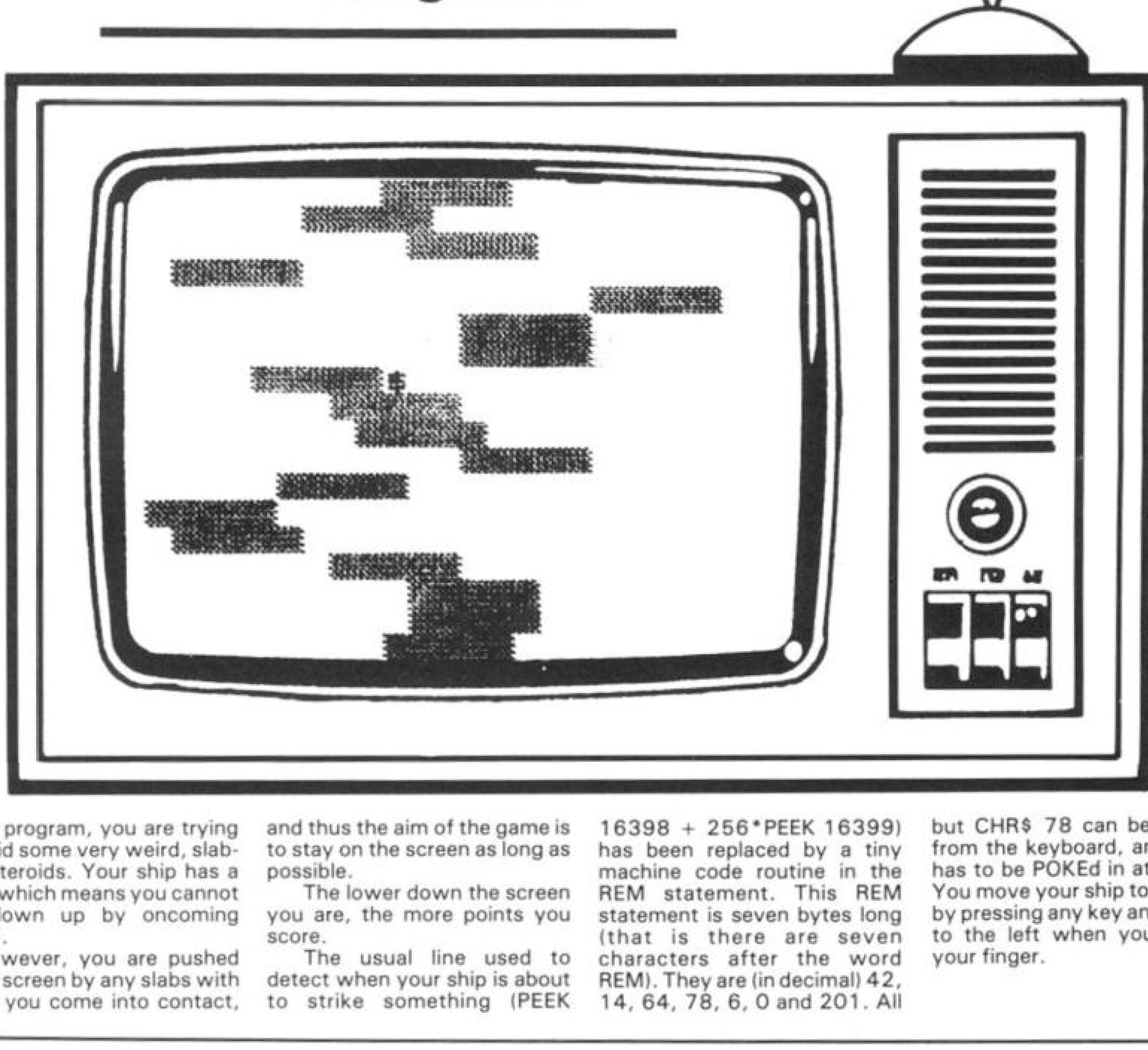
```
PRO POR  TIO NALSP A C INGPROPOR  TION AL SPACING
PROPOR  TIO NALS PACI N IGPRO PO   RTION SPACIN
P RO PORT  IONAL   SPAC ING   PROPORT  ION  ALSP
P R O P O R T I O N A L  SP AC I N G PROPORTIONALSPACING
P O R P OR  PORP O RPORPORPOR P O R P O R P O R
TION TION TIONAL SPACING PROPORTIONALPROPORTIONAL
PROPORTI  ONAL SPA C I N G PRO PORT ION AL  SPACI N G
P R O R T I O N A L P O R P RO   ING SPACING
PROPORTIONAL SPACING     PROPORTIONAL SPACING
     PROPORTI  ONAL SPACING   P R O T I O N A L S P A C
""""""""""""""""""""PROPORTIONAL SPACING""""""""""""""""""""
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
```

```
 10 LET B$=""
 20 LET A$=""
 30 PRINT A$
 40 PRINT AT 15,0;B$
 50 PAUSE 4E4
 60 POKE 16437,255
 70 LET C$=INKEY$
 80 CLS
 90 IF C$="<" THEN GOTO 130
 95 IF C$=">" THEN GOTO 340
100 IF CODE C$=118 THEN GOTO 15
0
110 LET A$=A$+C$
120 GOTO 30
130 LET A$=A$( TO LEN A$-1)
140 GOTO 30
150 IF LEN (B$+A$)+1>=32 THEN G
OTO 190
160 IF B$="" THEN GOTO 320
170 LET B$=B$+" "+A$
180 GOTO 20
190 LET A=32-LEN B$
200 IF A=0 THEN GOTO 310
210 FOR B=1 TO LEN B$
220 IF B$(B)<>" " THEN GOTO 300
230 LET B$=B$+" "
240 FOR C=LEN B$ TO B+1 STEP -1
250 LET B$(C)=B$(C-1)
260 NEXT C
270 LET B$(B)=" "
280 LET A=A-1
285 LET B=B+2
290 IF A=0 THEN GOTO 310
300 NEXT B
305 GOTO 210
310 LPRINT B$
320 LET B$=A$
330 GOTO 20
340 LPRINT B$
350 LPRINT
360 RUN
```


```
 10 LET B$=""
 20 LET A$=""
 30 PRINT A$
 40 PRINT AT 15,0;B$
 50 PAUSE 4E4
 60 POKE 16437,255
 70 LET C$=INKEY$
 80 CLS
 90 IF C$="<" THEN GOTO 130
 95 IF C$=">" THEN GOTO 340
100 IF CODE C$=118 THEN GOTO 15
0
110 LET A$=A$+C$
120 GOTO 30
130 LET A$=A$( TO LEN A$-1)
140 GOTO 30
150 IF LEN (B$+A$)+1>=32 THEN G
OTO 190
160 IF B$="" THEN GOTO 320
170 LET B$=B$+" "+A$
180 GOTO 20
190 LET A=32-LEN B$
200 IF A=0 THEN GOTO 310
210 FOR B=1 TO LEN B$
220 IF B$(B)<>" " THEN GOTO 300
230 LET B$=B$+" "
240 FOR C=LEN B$ TO B+1 STEP -1
250 LET B$(C)=B$(C-1)
260 NEXT C
270 LET B$(B)=" "
280 LET A=A-1
285 LET B=B+2
290 IF A=0 THEN GOTO 310
300 NEXT B
305 GOTO 210
310 LPRINT B$
320 LET B$=A$
330 GOTO 20
340 LPRINT B$
350 LPRINT
360 RUN
```

Surging away into space

Also from Tim Rogers comes this 1K 'arcade game'.



In this program, you are trying to avoid some very weird, slab-like asteroids. Your ship has a shield which means you cannot get blown up by oncoming 'slabs'.

However, you are pushed up the screen by any slabs with which you come into contact, and thus the aim of the game is to stay on the screen as long as possible.

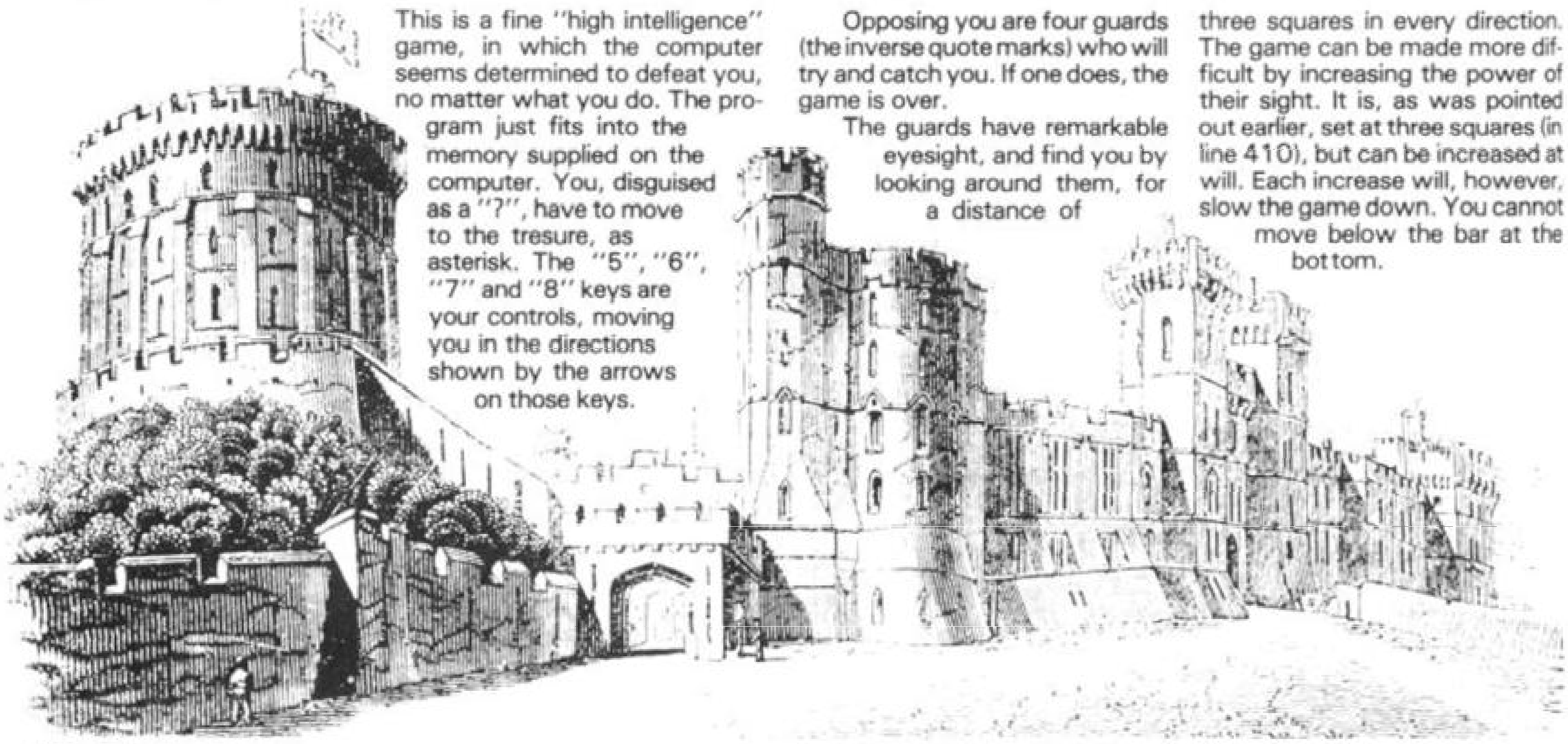
The lower down the screen you are, the more points you score.

The usual line used to detect when your ship is about to strike something (PEEK 16398 + 256*PEEK 16399) has been replaced by a tiny machine code routine in the REM statement. This REM statement is seven bytes long (that is there are seven characters after the word REM). They are (in decimal) 42, 14, 64, 78, 6, 0 and 201. All but CHR$ 78 can be entered from the keyboard, and so 78 has to be POKEd in at line 20. You move your ship to the right by pressing any key and it drifts to the left when you release your finger.

```
  1 REM E:RND?▞ TAN
  5 LET H=1
 10 POKE 16517,78
 15 LET S=H-H
 20 LET U=10
 25 LET T=20
 30 LET P=U
 40 PRINT AT U,P;" "
 50 LET P=P-H/H*(P>H/H)
 60 LET P=P+(INKEY$<>"")*2*(P<T
)
 65 SCROLL
 70 PRINT AT U,P;
 80 IF USR 16514=CODE "▒" THEN
LET U=U-H/H
 90 IF U=H-H THEN GOTO 200
100 PRINT AT U,P;"$"
110 PRINT AT CODE ")",RND*T;"▒▒▒
▒▒▒"
120 LET S=S+U
130 GOTO 40
200 IF H<S THEN LET H=S
210 PRINT AT U,P;"SURGE",S,H
220 PAUSE 4E4
230 GOTO 15
```

High security

Paul Holmes from Sutton Coldfield, author of the J.R.S. Software "Graphics Tool Kit", brings us a couple of clever games for the 16K ZX81.



This is a fine "high intelligence" game, in which the computer seems determined to defeat you, no matter what you do. The program just fits into the memory supplied on the computer. You, disguised as a "?", have to move to the tresure, as asterisk. The "5", "6", "7" and "8" keys are your controls, moving you in the directions shown by the arrows on those keys.

Opposing you are four guards (the inverse quote marks) who will try and catch you. If one does, the game is over.

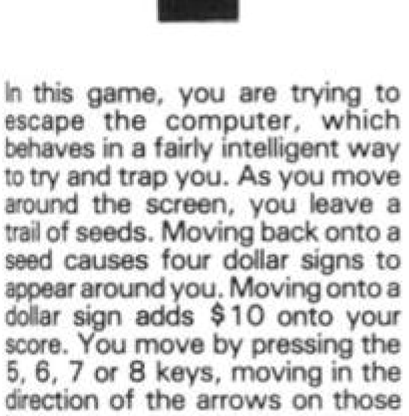
The guards have remarkable eyesight, and find you by looking around them, for a distance of three squares in every direction. The game can be made more difficult by increasing the power of their sight. It is, as was pointed out earlier, set at three squares (in line 410), but can be increased at will. Each increase will, however, slow the game down. You cannot move below the bar at the bottom.

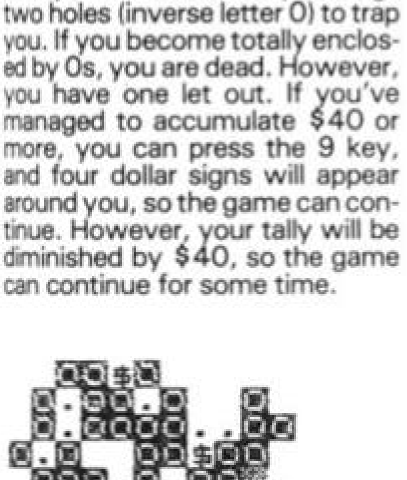
```
 30 DIM G(4)
 40 LET P=5
 50 LET Q=7
 80 LET C=610
 80 FOR Z=1 TO 13
 90 PRINT TAB 31;" "
100 NEXT Z
110 PRINT "--------------------------------"
115 PRINT AT 5,15;"*"
120 PRINT AT P,Q;"?"
130 PRINT AT 2,15;
140 GOSUB C
150 LET G(1)=A
160 PRINT AT 5,12;
170 GOSUB C
160 LET G(2)=A
190 PRINT AT 8,15;
200 GOSUB C
210 LET G(3)=A
220 PRINT AT 5,10;
230 GOSUB C
240 PRINT AT 0,0;
250 LET G(4)=A
280 INPUT A$
330 PRINT AT P,Q;" "
340 LET P=P-1*(A$="7")+1*(A$="6
")
350 LET Q=Q-1*(A$="5")+1*(A$="8
")
360 PRINT AT P,Q;"?"
400 FOR B=1 TO 4
405 POKE G(B),0
410 FOR A=1 TO 3
420 IF PEEK (G(B)-A)=15 THEN LE
T G(B)=G(B)-1
430 IF PEEK (G(B)+A)=15 THEN LE
T G(B)=G(B)+1
440 IF PEEK (G(B)+A*33)=15 THEN
LET G(B)=G(B)+33
450 IF PEEK (G(B)-A*33)=15 THEN
LET G(B)=G(B)-33
460 IF PEEK (G(B)-A*34)=15 THEN
LET G(B)=G(B)-34
470 IF PEEK (G(B)+A*34)=15 THEN
LET G(B)=G(B)+34
480 IF PEEK (G(B)-A*32)=15 THEN
LET G(B)=G(B)-32
490 IF PEEK (G(B)+A*32)=15 THEN
LET G(B)=G(B)+32
500 NEXT A
505 POKE G(B),139
510 NEXT B
550 PRINT AT P,Q;
560 GOSUB C
570 IF PEEK (A)=139 THEN STOP
580 IF P=5 AND Q=15 THEN GOTO 5
70
590 GOTO 260
600 STOP
610 LET A=PEEK 16398+PEEK 16399
*256
620 RETURN
670 PRINT "WELL DONE"
```

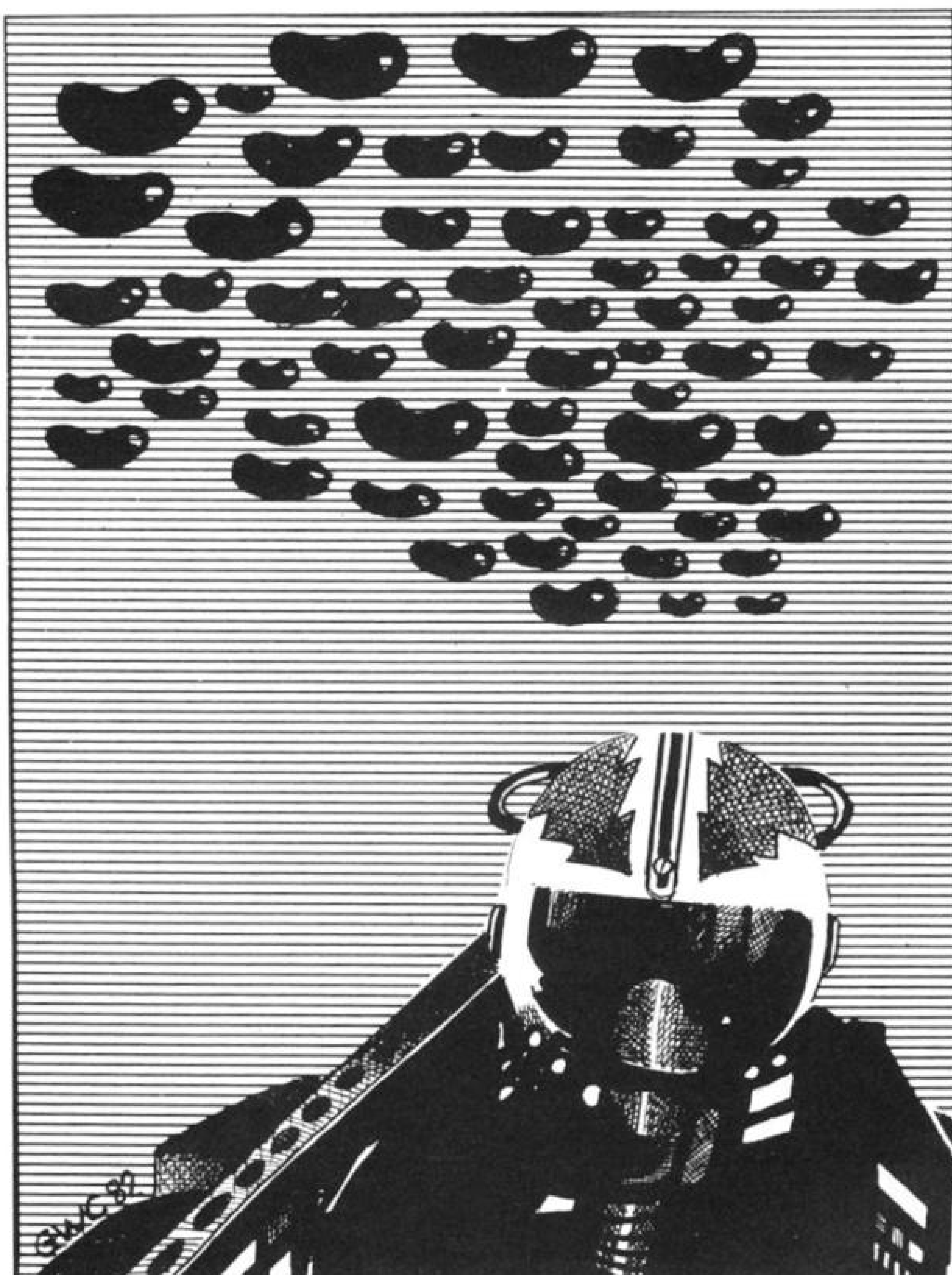
Magic Dollar Seeds

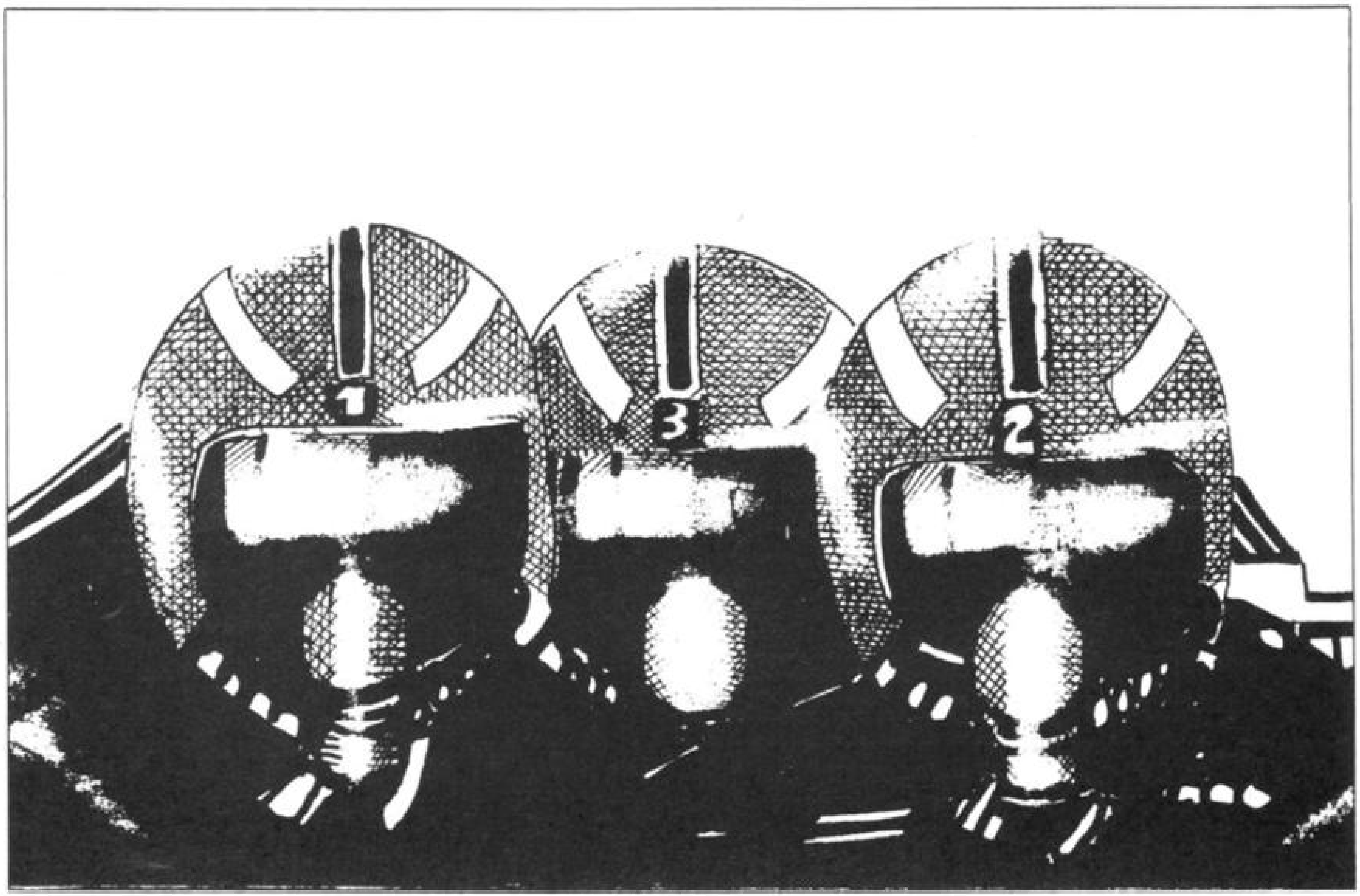
In this game, you are trying to escape the computer, which behaves in a fairly intelligent way to try and trap you. As you move around the screen, you leave a trail of seeds. Moving back onto a seed causes four dollar signs to appear around you. Moving onto a dollar sign adds $10 onto your score. You move by pressing the 5, 6, 7 or 8 keys, moving in the direction of the arrows on those keys.

The snag in all this is that each time you move, the computer digs two holes (inverse letter O) to trap you. If you become totally enclosed by Os, you are dead. However, you have one let out. If you've managed to accumulate $40 or more, you can press the 9 key, and four dollar signs will appear around you, so the game can continue. However, your tally will be diminished by $40, so the game can continue for some time.



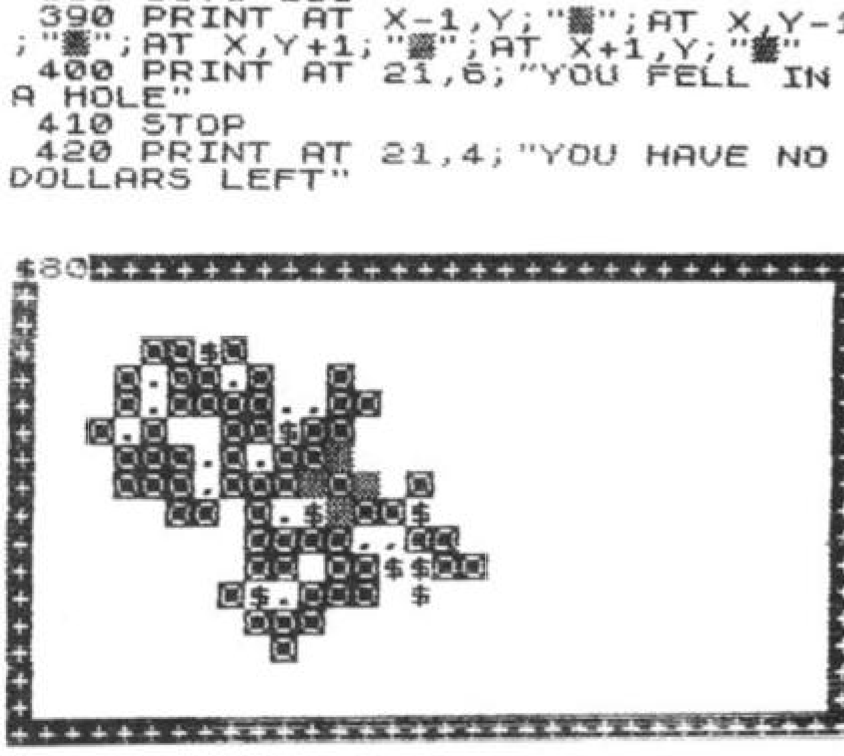






```
  10 LET S=0
  20 LET X=11
  30 LET Y=15
  40 PRINT "++++++++++++++++++++++++++++++++"
  50 FOR Z=1 TO 15
  60 PRINT "+";TAB 31;"+"
  70 NEXT Z
  80 PRINT "++++++++++++++++++++++++++++++++"
  90 PRINT AT X,Y;"+"
 100 PRINT AT 0,0;"$";S;TAB 10;"
READY"
 110 LET A$=INKEY$
 120 IF A$="" THEN GOTO 100
 130 PRINT AT 0,10;"++++++";AT X
,Y;"."
 140 IF A$<"5" OR A$>"9" THEN GO
TO 110
 150 IF A$="6" AND X<16 THEN LET
X=X+1
 160 IF A$="7" AND X>1 THEN LET

 150 IF A$="6" AND X<16 THEN LET
X=X+1
 160 IF A$="7" AND X>1 THEN LET
X=X-1
 170 IF A$="8" AND Y<30 THEN LET
Y=Y+1
 180 IF A$="5" AND Y>1 THEN LET
Y=Y-1
 190 IF A$="9" THEN LET S=S-40
 210 IF S<0 THEN GOTO 420
 220 PRINT AT X,Y;
 230 LET A$=CHR$ PEEK (PEEK 1639
8+PEEK 16399*256)
 240 IF A$="■" THEN GOTO 390
 250 IF A$="." THEN PRINT AT X-1
,Y;"$";AT X,Y-1;"$";AT X+1,Y;"$"
;AT X,Y+1;"$"
 260 IF A$="$" THEN LET S=S+10
 270 PRINT AT X,Y;"+"
 280 FOR Z=1 TO 2
 290 LET A=INT (RND*4+1)
 300 LET P=X
 310 LET Q=Y
 320 IF A=1 THEN LET P=X+1
 330 IF A=2 THEN LET P=X-1
 340 IF A=3 THEN LET Q=Y+1
 350 IF A=4 THEN LET Q=Y-1
 360 PRINT AT P,Q;"■"
 370 NEXT Z
 380 GOTO 100
 390 PRINT AT X-1,Y;"■";AT X,Y-1
;"■";AT X,Y+1;"■";AT X+1,Y;"■"
 400 PRINT AT 21,6;"YOU FELL IN
A HOLE"
 410 STOP
 420 PRINT AT 21,4;"YOU HAVE NO
DOLLARS LEFT"
```



Mastering machine code on your Spectrum

Toni Baker, author of 'Mastering Machine Code on Your ZX81', turns her attention to the Spectrum with this article, the first in a series designed to take you through machine code from its very beginnings to its ultimate conclusions.

Inside the Spectrum is a tiny little black box mystically referred to as a "Z80A". In fact the Z80A is the only part of the whole computer that actually does any thinking. To put it another way, the Z80A *is* the computer. The ROM is *not* a computer – the ROM just contains a computer *program*. The Z80A speaks a language we call MACHINE CODE. It does not speak BASIC. When you RUN a BASIC program what's really happening is that the Z80A is running a program in the ROM which tells it to look at what's written down in the RAM and then take appropriate action.

Machine code has variables just like in BASIC, but they're not quite as flexible. The registers are called A, B, C, D, E, H, and L, and they can only store integers from 0 to 255. It's easier to work in hexadecimal so I'll do that from the start – 00 means 0 and FF means 255. In general two symbols (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E or F) written next to each other means sixteen times the first digit, plus the second digit – leading zeroes are therefore optional – however DON'T waste your time converting things back to decimal all the time – you don't need to. 5A is obviously bigger than 3E because 5 is bigger than 3. In the same way D7 is a bigger number than AA.

It is not necessary to change the numbers into decimal first – it is better to get a kind of "feel" for the size of a number in hex without actually knowing what is is. After all, that's all we do in decimal isn't it? I bet you can't imagine a pile of (exactly) seventy three pennies.

A variable in machine code can therefore hold any number between 00 and FF. A machine code variable is called a REGISTER. There are no error traps in machine code, and so if you try to add up two numbers whose sum is more than FF you will get the wrong answer (in actual fact it will be 100 (hex) less than the real answer) – the last two digits will be the only ones that count. Take a look at this little segment of machine code:

LD A,9A – This is like a LET statement. Register A now holds the number 9A.
ADD A,88 – In machine code you can only do one thing at a time – you cannot, eg. say LD A,9A + 88 as you could in BASIC. What value does the A register now contain? The answer is 22. Try to do the adding up in hex: A plus 8 equals 2 carry 1; 9 plus 8 plus the carry is also 2 carry 1; this carry is "lost".

Registers can also be used in pairs. The only combinations allowed are BC, DE, and HL. If B contains 61 and C contains A7 then we say that BC contains 61A7. This is a four digit hexadecimal number. Its size is intuitively just a bit bigger than 6000, and a lot less than 7000. Similarly, if HL contains 1234 we say that H contains 12 and L contains 34.

```
  10 DEF FN k(x)=CODE "012345678
9000000000:;<=>?0000000000000000000
000000000:;<=>?"(CODE a$(x)-47)-4
8
  20 LET a$=""
  30 INPUT x
  40 IF a$="" THEN INPUT a$
  50 POKE x,16*FN k(1)+FN k(2)
  55 PRINT a$( TO 2);"  ";
  60 LET a$=a$(3 TO )
  70 LET x=x+1
  80 GO TO 40
```

Figure 1. HEXLD.

How do we actually USE machine code?

When the ZX83 comes out, hopefully there will be a few buttons marked with machine code instructions. Until that happens we unfortunately have to do some translating. Each machine code instruction has a number – a sort of index. Instruction number one is LD BC, – something like a LET statement in BASIC. All the Z80A needs is a list of numbers. Whenever it comes across the number 01 it knows it has to carry out the operation LD BC., It also of course expects a four digit number next so that it knows what to load BC *with*. This index number is called a HEX CODE.

The words we use for the instructions are sometimes called OP CODES (Operation Codes). For every OPcode there is a HEXcode, and for every HEXcode there is an OPcode.

The computer needs the HEXcodes in its programs. Humans on the other hand find it easier to use the OPcodes. When writing down a machine code program on a piece of paper we usually then write BOTH versions next to each other – like this:

```
C9                 RET
```

Here C9 is HEXcode which the computer will understand. RET is our way of writing it. RET means RETURN; either "Return to BASIC" as we shall use very shortly, or "Return from a subroutine" which I shall cover in a later article.

Every machine code program you write must end with a RET instruction.

The meaning of USR

USR is a function in BASIC – it's very much like a cross between a GOSUB statement and user defined (numerical) function. It looks very much like a function in appearance: USR X has the same "shape" as SIN X or INT X, and can be used in exactly the same circumstances. But if SIN X equals the height of a sine wave at position X, and if INT X equals X with all of its decimals banished, what number does USR X give us? ANSWER: USR X gives us the value of the BC register! A machine code program is run every time USR is used, and the number of variable or whatever after the word USR must be the address of the start of a program written in machine code. For example, consider this machine code program:

```
010000             LD BC,0000
C9                 RET
```

If X was the address of the "01" in the above, then what number would USR X give us? RET, remember, means return to BASIC, and so BC ends up as zero. In this case USR X would give us a value of zero, so PRINT USR X would print 0, and LET Y=USR X would assign Y with zero, and so on.

Our next problem is how do we get the machine code into the computer in the first place? The only way to do it is with a BASIC program. Take a look at the program in Fig. 1. It's called HEXLD, and I shall explain what it does and how it works.

The first line is a user-defined function which turns a string character into a number. Its effect is to turn "0" into 0, "1" into 1, and so on until "9" which becomes 9. In addition "A" becomes ten, "B" becomes eleven, and so on up to "F" which becomes fifteen. Small letters are

```
  10 PRINT " LIST ";
  20 GO SUB 8000
  30 RANDOMIZE USR 65055
  40 STOP
 100 PRINT "WRITE ";
 110 GO SUB 8000
 120 INPUT A$: PRINT ;
 130 RANDOMIZE USR 65152
 140 PRINT A$
 150 GO TO 120
 200 PRINT "INSERT ";
 210 GO SUB 8000
 220 INPUT A$: PRINT ;
 230 RANDOMIZE USR 65109
 240 PRINT A$
 250 GO TO 220
 300 GO SUB 320
 310 STOP
 320 PRINT " DELETE ";
 330 GO SUB 8000
 340 PRINT "       TO ";
 350 GO SUB 9000
 360 RANDOMIZE USR 65235
 370 RETURN
 400 SAVE "HEXLD 3" LINE 460
 410 SAVE "HEXLD 3 MC"CODE 65016
353
 420 SAVE " "CODE FN P(65016),FN
P(65024)-FN P(65016)+1
 430 VERIFY ""
 440 VERIFY ""CODE
 450 VERIFY ""CODE : STOP
 460 BORDER 0: INK 7: PAPER 0: F
LASH 0: BRIGHT 0
 470 CLEAR 65015
 480 LOAD ""CODE : LOAD ""CODE :
STOP
 500 PRINT "BEGIN ";
 510 GO SUB 8000
 520 RANDOMIZE USR 65264
 530 STOP
 600 PRINT "REPLACE ";
 610 GO SUB 330
 620 GO TO 220
 700 PRINT " RUN ";
 710 GO SUB 8000
 720 RANDOMIZE USR FN P(65018)
 730 STOP
 800 PRINT " COPY ";: GO SUB 800
0
 810 PRINT "         ";: GO SUB 900
0
 820 PRINT "    TO ";
 830 INPUT "ADDRESS ";a$
 840 PRINT "ADDRESS ";a$
 850 POKE 65022,16*FN k(a$,3)+FN
 k(a$,4)
 860 POKE 65023,16*FN k(a$,1)+FN
 k(a$,2)
 870 RANDOMIZE USR 65275
8000 INPUT "ADDRESS ";a$
8010 PRINT "ADDRESS ";a$
8020 POKE 65018,16*FN k(a$,3)+FN
 k(a$,4)
8030 POKE 65019,16*FN k(a$,1)+FN
 k(a$,2)
8040 RETURN
9000 INPUT "ADDRESS ";a$
9010 PRINT "ADDRESS ";a$
9020 POKE 65020,16*FN k(a$,3)+FN
 k(a$,4)
9030 POKE 65021,16*FN k(a$,1)+FN
 k(a$,2)
9040 RETURN
9050 DEF FN k(x$,y)=CODE "012345
6789000000:;<=>?000000000000000
00000000000:;<=>?"(CODE x$(y)-47
)-48
9060 DEF FN h(x$)=4096*FN k(x$,1
)+256*FN k(x$,2)+16*FN k(x$,3)+F
N k(x$,4)
9070 DEF FN k$(x,y)="0123456789A
BCDEF"(INT (x/16↑y)-16*INT (x/16
↑(y+1))+1)
9080 DEF FN h$(x)=FN k$(x,3)+FN
k$(x,2)+FN k$(x,1)+FN k$(x,0)
9090 DEF FN P(X)=PEEK X+256*PEEK
 (X+1)
```

Fig 2 (Above) The listing of HEXLD 3.

Fig 3 (Below) The code for HEXLD 3.

```
FE03 E6
FE04 F0
FE05 1F
FE06 1F
FE07 1F
FE08 1F
FE09 C6
FE0A 30  0
FE0B FE
FE0C 3A  :
FE0D 38  8
FE0E 02
FE0F C6
FE10 07
FE11 D7
FE12 F1
FE13 E6
FE14 0F
FE15 C6
FE16 30  0
FE17 FE
FE18 3A  :
FE19 38  8
FE1A 02
FE1B C6
FE1C 07
FE1D D7
FE1E C9
FE1F 2A  *
FE20 00
FE21 FE
FE22 22  "
FE23 FC
FE24 FD
FE25 54  T
FE26 5D  ]
FE27 2A  *
FE28 FA
FE29 FD
FE2A A7
FE2B ED
FE2C 52  R
FE2D 19
FE2E D0
FE2F 7C  |
FE30 CD
FE31 02
FE32 FE
FE33 7D  }
FE34 CD
FE35 02
FE36 FE
FE37 3E  >
FE38 20
FE39 D7
FE3A 7E  ~
FE3B CD
FE3C 02
FE3D FE
FE3E 3E  >
FE3F 20
FE40 D7
FE41 7E  ~
FE42 FE
FE43 20
FE44 38  8
FE45 05
FE46 FE
FE47 A5
FE48 30  0
FE49 01
FE4A D7
FE4B 3E  >
FE4C 0D
FE4D D7
FE4E 23  #
FE4F 22  "
FE50 FA
FE51 FD
FE52 18
FE53 D6
FE54 00
FE55 2A  *
FE56 4B  K
FE57 5C  \
FE58 23  #
FE59 4E  N
FE5A 23  #
FE5B 46  F
FE5C CB
FE5D 28  (
FE5E C5
FE5F 19
FE60 20
FE61 02
FE62 CF
FE63 14
FE64 C5
FE65 2A  *
FE66 00
FE67 FE
FE68 ED
FE69 5B  [
FE6A FA
FE6B FD
FE6C A7
FE6D ED
FE6E 52  R
FE6F 23  #
FE70 44  D
FE71 4D  M
FE72 E1
FE73 ED
FE74 5B  [
FE75 00
FE76 FE
FE77 19
FE78 22  "
FE79 00
FE7A FE
FE7B EB
FE7C ED
FE7D B0
FE7E 00
FE7F 00
FE80 2A  *
FE81 4B  K
FE82 5C  \
FE83 23  #
FE84 4E  N
FE85 23  #
FE86 46  F
FE87 CB
FE88 28  (
FE89 CB
FE8A 19
FE8B 28  (
FE8C D5
FE8D ED
FE8E 5B  [
FE8F FA
FE90 FD
FE91 7A  z
FE92 CD
FE93 02
FE94 FE
FE95 7B  {
FE96 CD
FE97 02
FE98 FE
FE99 3E  >
FE9A 20
FE9B D7
FE9C 23  #
FE9D 7E  ~
FE9E FE
FE9F 40  @
FEA0 38  8
FEA1 04
FEA2 E6
FEA3 DF
FEA4 D6
FEA5 07
FEA6 87
FEA7 87
FEA8 87
FEA9 87
FEAA D5
FEAB 57  W
FEAC 23  #
FEAD 7E  ~
FEAE FE
FEAF 40  @
FEB0 38  8
FEB1 04
FEB2 E6
FEB3 DF
FEB4 D6
FEB5 07
FEB6 D6
FEB7 30  0
FEB8 82
FEB9 D1
FEBA 12
FEBB 13
FEBC ED
FEBD 53  S
FEBE FA
FEBF FD
FEC0 E5
FEC1 2A  *
FEC2 00
FEC3 FE
FEC4 ED
FEC5 52  R
```

```
FEC6  E1          FEE4  5B  [       FF01  FD          FF1E  CF          FF3C  02
FEC7  30  0       FEE5  FA          FF02  A7          FF1F  08          FF3D  FE
FEC8  04          FEE6  FD          FF03  ED          FF20  CD          FF3E  78  x
FEC9  ED          FEE7  ED          FF04  52  R       FF21  02          FF3F  D7
FECA  53  S       FEE8  B0          FF05  44  D       FF22  FE          FF40  7C  |
FECB  00          FEE9  1B          FF06  4D  M       FF23  3E  >       FF41  CD
FECC  FE          FEEA  1B          FF07  03          FF24  20          FF42  02
FECD  0B          FEEB  ED                            FF25  D7          FF43  FE
FECE  78  x       FEEC  53  S       FF08  2A  *       FF26  78  x       FF44  78  x
FECF  B1          FEED  00          FF09  FE          FF27  CD          FF45  D7
FED0  20          FEEE  FE          FF0A  FD          FF28  02          FF46  7D  }
FED1  CA          FEEF  C9          FF0B  EB          FF29  FE          FF47  CD
FED2  C9          FEF0  2A  *       FF0C  A7          FF2A  3E  >       FF48  02
FED3  2A  *       FEF1  FA          FF0D  ED          FF2B  20          FF49  FE
FED4  00          FEF2  FD          FF0E  52  R       FF2C  47  G       FF4A  3E  >
FED5  FE          FEF3  22  "       FF0F  19          FF2D  D7          FF4B  0D
FED6  ED          FEF4  F8          FF10  38  8       FF2E  79  y       FF4C  D7
FED7  5B  [       FEF5  FD          FF11  04          FF2F  CD          FF4D  E1
FED8  FC          FEF6  2B  +       FF12  ED          FF30  02          FF4E  7C  |
FED9  FD          FEF7  22  "       FF13  B0          FF31  FE          FF4F  CD
FEDA  D5          FEF8  00          FF14  CF          FF32  78  x       FF50  02
FEDB  A7          FEF9  FE          FF15  08          FF33  D7          FF51  FE
FEDC  ED          FEFA  C9          FF16  09          FF34  7A  z       FF52  7D  }
FEDD  52  R       FEFB  ED          FF17  2B  +       FF35  CD          FF53  CD
FEDE  44  D       FEFC  5B  [       FF18  EB          FF36  02          FF54  02
FEDF  4D  M       FEFD  FA          FF19  09          FF37  FE          FF55  FE
FEE0  03          FEFE  FD          FF1A  2B  +       FF38  78  x       FF56  CF
FEE1  E1          FEFF  2A  *       FF1B  EB          FF39  D7          FF57  14
FEE2  23  #                         FF1C  ED          FF3A  7B  {
FEE3  ED          FF00  FC          FF1D  B8          FF3B  CD
```

Fig 3 Cont. The code for HEXLD 3.

also taken into account and so "a" also becomes ten, and so on up to "f" which gives fifteen just as if it were a capital. The rest of the program is your HEX LOADER.

To use the program type RUN, then input a (decimal) address. Input 24576 here (for no other reason than the fact that in hex 24576 is written as 6000). Now all you need to type in is your machine code. Type in "010000" and then "C9". To stop the program type in just a newline – this will cause error code 3. You now have a machine code program. Type PRINT USR 24576 to see if it gives zero as it should. If you want to see what you're doing change line 40 to read IF a$ = "" THEN INPUT a$: PRINT a$

For advanced programmers . . .

Figures 2 and 3 give a machine code editing program called HEXLD 3. You can load it into the computer using HEXLD as above. Its purpose is to allow you to construct and edit other programs in machine code. To avoid confusion the hex given in Fig. 3 is called the "object program" – the machine code you will use it to edit is referred to as the "subject program". You should not attempt to use HEXLD 3 to edit itself.

If you are using a 16K machine instead of a 48K machine you must subtract 32768 from each address used in the BASIC, and you must change each address referred to in the machine code which begins with F into the corresponding address beginning with 7.

The features of the program are as follows:

The arrangement of the variables and machine code routines of HEXLD 3

16K	48K		
7DF8	FDF8	BEGIN	Points to the first byte of the subject program.
7DFA	FDFA	ADDRESS	)
7DFC	FDFC	ADD2	)Parameters used by HEXLD 3.
7DFE	FDFE	ADD3	)
7E00	FE00	LIMIT	Points to the first byte after the subject program.
7E02	FE02	HPRINT	Subroutine to print the contents of the A register in hexadecimal.
7E1F	FE1F	HLIST	Lists subject program in hexadecimal.
7E55	FE55	INSERT	Inserts additional bytes into subject program.
7E80	FE80	WRITE	Overwrites subject program with new code.
7ED3	FED3	DELETE	Deletes bytes from subject program.
7EF0	FEF0	BEGINMC	Sets BEGIN and LIMIT ready for creating new subject program.
7EFB	FEFB	HCOPY	Overwrites subject program with bytes copied from elsewhere.
7F20	FF20	BREAKP	Break point routine.
7F58	FF58		Next spare byte – User defined graphics may begin here.

Fig. 4

RUN	*allows you to list machine code in hex from any (hex address).*
RUN 100	*allows you to write your own machine code as in HEXLD above.*
RUN 200	*allows you to insert bytes of machine code between existing bytes.*
RUN 300	*allows you to delete bytes of machine code, closing up the gap which they occupied.*
RUN 400	*SAVEs first the BASIC, then the object program, then the subject program, then verifies all three.*
RUN 500	*Initially assigns variables used by this program. RUN 500 must be used only if you are creating a new program from scratch.*
RUN 600	*Equivalent to DELETE followed by INSERT at the same address.*
RUN 700	*allows you to RUN machine code from any address.*
RUN 800	*allows you to copy blocks of machine code from one address to another.*
FN H(string)	*changes hex to decimal, eg FN H ("002A") = 42*
FN H$ (number)	*changes decimal to hex, eg FN H$(42) = "002A"*
FNP (address	*equivalent to PEEK (address) + 256* PEEK (address + 1)*

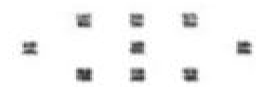
I am sorry if there is insufficient space to list the machine code for HEXLD 3 in full. You may like to translate it for yourself as an excercise if you are sufficiently masochistic.

The Secret of LIFE

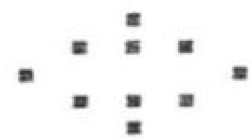
We set you a problem, and show one way to solve it

```
 10 FAST
 20 DIM E(8)
 30 LET A$="-+=£▒▀ ▘ "
 40 FOR A=1 TO 8
 50 LET E(A)=CODE H$(H)-11
 60 NEXT A
 70 DIM A(120)
 80 DIM L(120)
 90 LET A$="64,55.65.75.76,46,5
6,76,86,"
100 FOR A=1 TO LEN A$ STEP 3
110 LET A(VAL A$(A TO A+1))=1
120 LET L(VAL A$(A TO A+1))=1
130 NEXT A
140 LET GENERATION=0
145 SLOW
150 GOTO 310
160 LET GENERATION=GENERATION+1
170 FOR U=0 TO 9
180 FOR B=1 TO 9
190 LET F=U+10*B+2
200 LET H=0
210 FOR T=1 TO 8
220 LET H=H+A(F+E(T))
230 NEXT T
240 IF A(F)=1 AND H<>3 AND H<>2
THEN LET L(F)=0
250 IF A(F)=0 AND H=3 THEN LET
L(F)=1
260 NEXT B
270 NEXT U
275 SLOW
280 FOR M=1 TO 100
290 LET A(M)=L(M)
300 NEXT M
310 PRINT AT 5,0;
320 FOR U=0 TO 9
330 PRINT TAB 3;
340 FOR B=0 TO 9
350 LET F=U+10*B+1
360 PRINT CHR$ A(F);" ";
370 NEXT B
380 NEXT U
390 PRINT AT 3,10;"GENERATION "
;GENERATION
400 FOR G=1 TO 100
410 NEXT G
420 FAST
430 GOTO 160
```

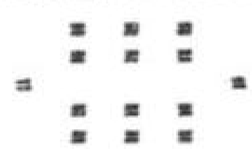
GENERATION 0



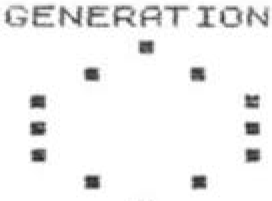
GENERATION 1



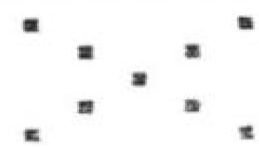
GENERATION 2



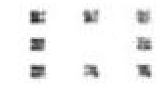
GENERATION 3



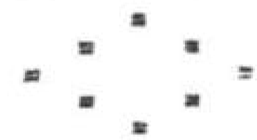
GENERATION 0



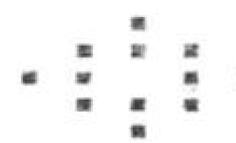
GENERATION 1



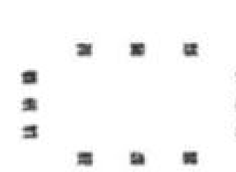
GENERATION 2



GENERATION 3



GENERATION 4



The game of LIFE was invented by John Conway of Cambridge University in October, 1970. It simulates the birth, death and growth of cells in a closed colony.

Before the state of a cell for the 'next generation' (a generation is a complete check, and reprint of the grid upon which the colony lives) is determined, it must be compared with the eight surrounding cells. If there are two or three occupied cells around the one being checked, and the one being checked is occupied, there is no change; it survives till the next generation. If there are three and only three occupied cells, and the cell being checked is empty, a cell is 'born' there in the next generation. If there are four or more neighbours, the cell being checked 'dies', that is, is emptied in the next generation.

That is almost all the information you need to construct a game of LIFE from first principles. There is just one more thing – the rules are applied all over the grid at once, so you need one array to hold the current generation, and another to hold the new generation, so that changes for the next generation do not effect cells which have not yet been checked in the present generation. Set up a 10 x 10 grid, and try and work out a program to (a) place some cells on it; (b) check each of these cells in turn in accord with Conway's laws, and then update a reference array; (c) copy the reference array into a 'print out' array; and (d) print out the colony and start again.

Here's one way of doing it, which uses two 'data' statements in the form of strings which are accessed element by element. A$ in line 30 contains information regarding the numerical relationship of cells to each other (eg + 1 is one to the right, – 1 is one to the left and so on). A$ in line 90 is the position of the starting cells, when the grid is numbered one to 100. Line 30 contains the following: minus sign, plus sign, equals sign, pound sign, graphic from the S key, graphic from the 2 key, graphic from the 1 key, space.

Note that there is a comma after the last element within A$ in line 90. This is needed for the "data" routine to work.

Other starting colonies you can try:

BEEHIVE: 45, 45, 46, 64, 65, 66, 74, 76, 85

CROSS: 43, 47, 54, 56, 65, 74, 76, 83, 87

MOBIUS: 23, 24, 25, 33, 34, 35, 43, 44, 45, 56, 57, 58, 66, 67, 68, 76, 77, 78

RUSSIAN: 33, 34, 35, 36, 37, 38, 47, 56, 65, 74, 83, 84, 85, 86, 87, 88

FLAME: 16, 26, 36, 46, 51, 52, 53, 54, 55, 56, 57, 58, 59, 66, 76, 86, 96

The ideal school computer

David Valentine looks at the ZX81 and asks if it really is the best machine for schools to use, even at £70.

It is easy to see why the ZX81 is becoming so popular in schools. It is a friendly looking computer, not much larger than the now commonplace calculator and it has rapidly become a familiar piece of technology due to the saturation advertising of Sinclair. It is easily available through outlets such as W.H. Smiths, Griffin and George (school equipment suppliers) as well as, of course, mail order. There can be very few people who have not seen at least a picture of one and many are purchased out of simple curiosity. It is hardly surprising then that just as the radio, the television and the video recorder have been taken up by schools, so has the ZX81.

Is it then the ideal compter for use in a school? The answer to this must be a very guarded 'maybe'. Certainly I myself use more than one on a regular basis with a greeat deal of success, but this is only after meeting and overcoming a number of problems. I have also seen it dismissed as a toy not worthy of serious consideration.

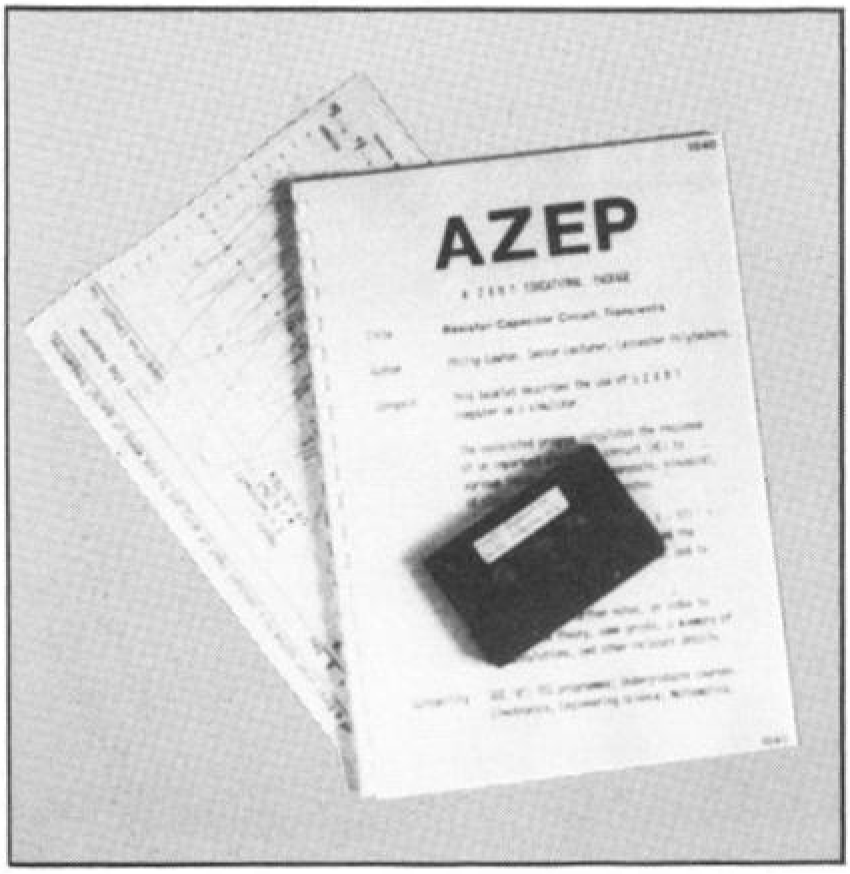
What then are the points in its favour? The major factor has to be cost. If the school has very limited funds or if it wants to afford a number of machines then the ZX81 wins hands down. It is worth remembering however, that no matter how good the value for money, if a device is difficult to use then it is not necessarily such a bargain. A ZX81 is inexpensive enough to allow home at weekends to suitable pupils and is of course easily carried (ever tried stuffing a PET into your satchel?). The standard keyboard is proof against having liquids spilled on it and having pencils stuffed down it, both important factors in a junior school. Despite its size it is certainly powerful enough for most conceivable applications in a junior school and for many in a secondary school. The main limitation is the tiny amount of memory available on the standard model and although some very clever programs have been written for the 1K version, I have found that they tend to be not very 'user friendly' in that there is little room for on screen explanation or error trapping and the use of graphics tends to be restricted.

What are the drawbacks to its usage and can they be overcome?

As I have already mentioned it will soon become apparent to a serious user that a memory expansion is required and it is here that we meet one of the major drawbacks of the ZX81. As you are no doubt aware the standard 16K add on memory pack is simply pushed onto the exposed part of the printed circuit board at the back of the computer. This connection has rapidly achieved notoriety for being amazingly unstable. In short, it wobbles and can often lead to loss of program. I originally used a wide rubber band to hold my RAM pack in place but I have recently acquired a device known as a 'WOBBLEQUITS'. This is a shaped piece of sprung steel which acts as both a base for the computer making it less liable to slide around, but also it holds the RAM pack tightly against the back of the computer. It has been found that some of the smaller 16K RAM packs currently available tend to be less prone to moving around.

In Schools, equipment must be both reliable and safe. I have been very happy that there is no danger to prying fingers from the ZX81, particularly when the RAM pack is covering up the exposed part of the board, however I am not so happy that the power supply lead is very easily pulled from its socket. A careful teacher will obviously place the leads in such a position that they are away from the children, however I have seen the power and the cassette leads pulled out on a number of occasions purely by accident as a result of over enthusiasm. There are a number of ways of overcoming this. One crude





method is to simply tape the plug in place with insulating tape. Another more sophisticated method is to make a wooden surround for the ZX81, boxing the leads on the left hand side. The power and the cassette leads are then replaced on the side of the box by more stable plugs such as 'BNC' types, easily available from any component shop or R.S. components for example. The existing leads are then cut, they are then soldered onto the back of the new plug. This short lead is now replugged into the ZX81 and can now be boxed in. Each long lead now has the 'BNC' male end soldered on and can then be plugged into the side of the box. As this does not affect the computer in any way it should not invalidate the guarantee. The leads can, of course, be soldered in directly onto the board, but as this involves opening up the ZX81 it should only be undertaken by someone who knows what he or she is doing as it is likely to invalidate the guarantee.

Much has been written concerning the unreliability of the LOADing system. Many is the time I have seen eagerness in the classroom turn to cynicism and boredom as a teacher has tried to load a program for the nth time, trying "just one more volume setting". Again, an experienced teacher will have experimented with all possible settings and will instinctively know if a program is loading. It is always a good idea to have loaded up any programs before the start of the lesson anyway!

The keyboard has come in for a lot of criticism, however I feel that a lot of the criticisms are invalid in schools. Pupils are not touch typists and tend to adapt very quickly to the idea of stroking the keys. The keyboard is rather cluttered but again I have been pleasantly surprised at the speed with which children have learned where each function is. They pick that sort of thing up much more quickly than most adults! I must add, however, that the lack of lower case letters must limit its use in an infant school. Teachers of reading have thrown up their hands in horror at the thought of having to use upper case all of the time!

I have spoken to people who are concerned with 'software libraries' for Local Education Authorities and have asked them why they are reticent to support the ZX81 and it is largely down to the drawbacks I have mentioned. Personally I feel such attitudes are rather shortsighted — the ZX81 has a lot to offer a School, however, unreliable technology is rapidly classed as a gimmick and is soon passed over by an experienced teacher who has 'seen it all before'.

Conclusion

A powerful easy to use computer which is without doubt a very suitable machine for anyone who wants to learn or teach the fundamentals of computing for the minimum outlay. For more serious and extended applications in schools some thought has to be given to make it as reliable as possible if pupils are not to become frustrated in using it.

MUSE winners announced

Final awards have now been made in the special ZX81 software award scheme, organised last November by MUSE, the educational computing association, to encourage the production of learning programs, and sponsored by Sinclair Research.

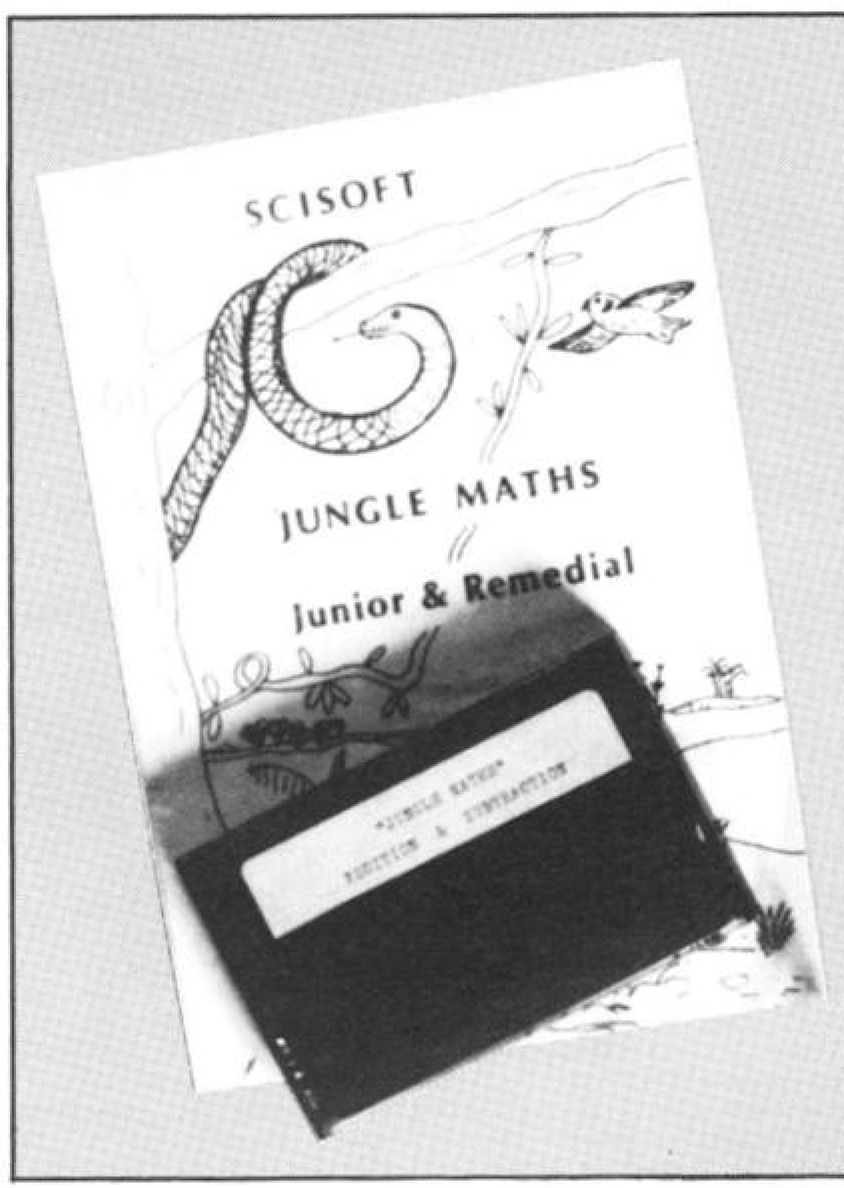
Well over 100 entries — "a most gratifying response", according to the organisers — were received. Many have been accepted into the ZX81 section of the MUSE software library, which with nearly 50 programs is now the largest section.

Eric Deeson, Educational ZX User Group organiser, and MUSE software librarian, Charles Sweeten, eventually decided to award prizes in only five of the six possible categories, "reflecting", said Eric, "less poor quality than an over-preponderance of science and maths material".

Prizes of Sinclair ZX Printers went to Dave Fisher of Coventry in the primary maths/science section for 'Bomber'; to Charles Rowbotham of Manchester under other primary for 'Sentry'; under secondary maths/science to John McMullan of Stechford for 'Forensic'; under other secondary to Richard Marriot of Kenilworth for 'Bigspell', and under other to Ian Souter of Tunbridge Wells for 'TLOG'. No award was made in the administration category.

Announcing the results Eric Deeson expressed MUSE's thanks "to Sinclair for making the awards and covering the costs of the exercise, and to the assessors for their always considerate and definitive reports".

MUSE is a national organisation for co-ordinating activity in primary and secondary schools, teacher training institutions, colleges and other establishments with an interest in the use of mini and microcomputers in any subject area of education. For further information on MUSE please contact Bob Trigger, MUSE Freepost, Bromsgrove B61 0JT, Worcs.



Enough to send you up a tree

James Walsh enters the numerical jungle

Come on, who are they trying to kid. Firstly what has the jungle got to do with maths, secondly, maths cannot be fun... can it? Well at least it is a little less mind bending than 'O' level chemistry or Undergraduate electronics.

Jungle maths is written for 'Juniors and Remedials', presumably between the ages of about six and thirteen. The jungle idea is that you are situated in a graphically represented jungle and have to get back to base. If you get a question right then you move on one place. If you get it wrong you lose one of your five lives in one of four weird and wonderful ways.

Each way is very well represented by some advanced moving graphics. If you take too long to answer the question, you see a full screen picture of "yourself" sink into quick-sand, and you lose a life.

On side one you have addition and subtraction with these choices; whole numbers, decimals, minus values, time-limit, size of number. On side two there are multiplication and division with the same choices.

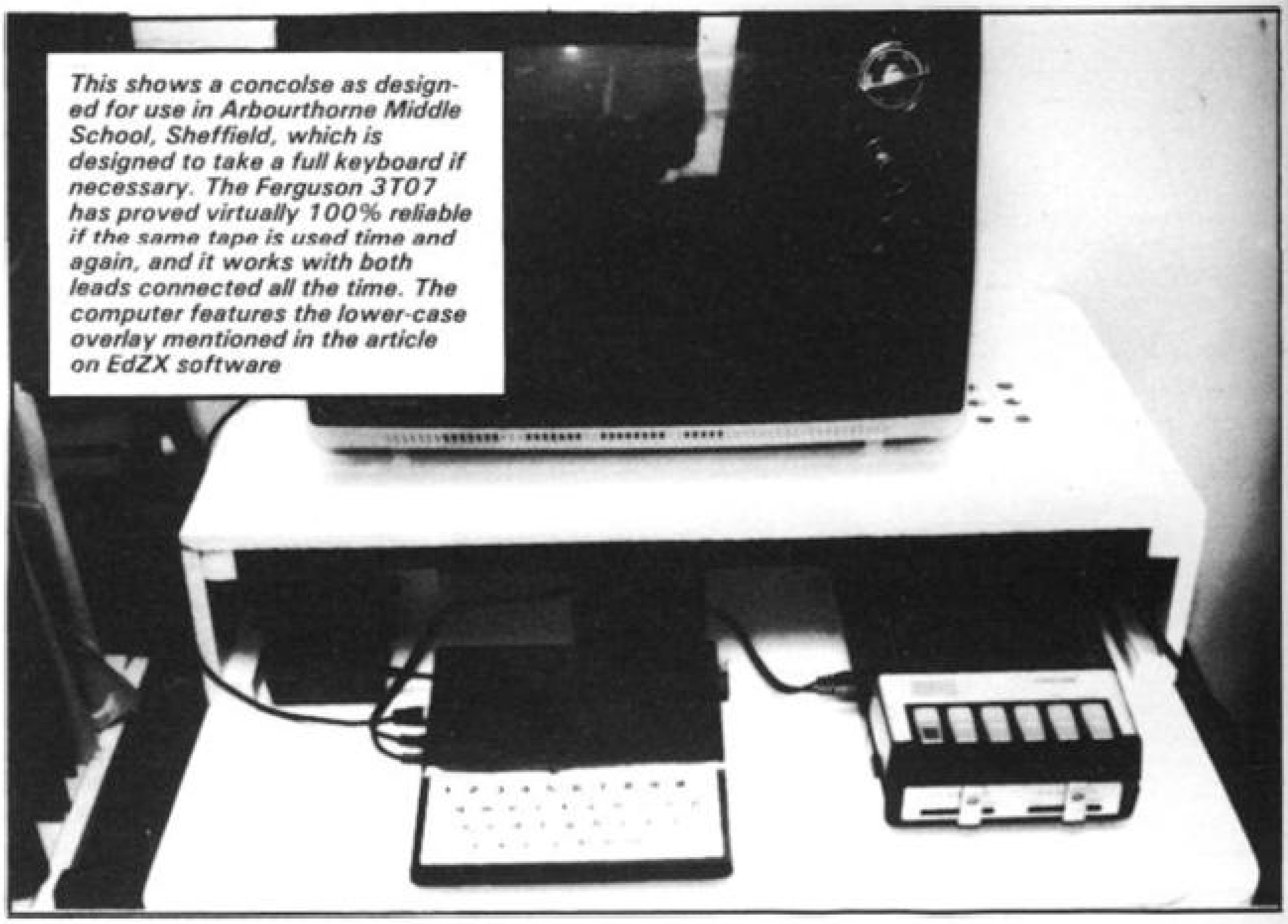
Conclusion

The documentation is good and gives the teacher quite a lot of help. Overall, this package is excellent. I have seen some educational packages on more expensive computers such as the MZ80K and the Apple and this is the first time I have seen a ZX81 educational program with graphics anywhere near as good. Personally my only reservation was that I was tempted to get answers wrong in order to see myself being eaten by piranhas or fall into a deep pit. Due to the versatility of the program and the number of choices, it can be used by anyone from the bright six-year-old to the remedial thirteen-year-old.

Well done SCISOFT, this is very good indeed and I recommend it.

Making connections

James Walsh foresakes the trees for the p.c.b.'s



This shows a concolse as designed for use in Arbourthorne Middle School, Sheffield, which is designed to take a full keyboard if necessary. The Ferguson 3T07 has proved virtually 100% reliable if the same tape is used time and again, and it works with both leads connected all the time. The computer features the lower-case overlay mentioned in the article on EdZX software

From 'O' level to 'A' level in one giant leap. In the last edition I looked at three 'O' level revision packages, I have now made a rather rapid jump up to 'A' Level with Philip Lawton's "Resistor — Capacitor Transients" package. This particular package is part of a series of program packages and video cassettes produced and marketed by Mr. Lawton. To put you in the picture, this package is aimed at GCE "A" students, TEC programmes and Undergraudate courses in subjects such as electronics, engineering, science and mathematics.

Together with the program, which incidently loaded first time, comes a hefty 26 page booklet aimed at giving the tutors themselves an idea of what the package is all about. The documentation is thorough, well thought out and constitutes a large proportion of the value of this package, hence I intend to spend a little more time than usual looking through the contents of the documentation.

The first couple of pages give a general outline of the contents, suitable courses and notes on the equations. Page three gives an index to video recording if you decide to record it (we will come onto this later). Pages 4 through to 9 give a basic run down of the program which you will need altering for adaption to other computers, such as the PET.

The remaining 17 pages are dedicated to a script for use either as a guide for the lecturer when using the program with his/her students or as a script for a video recording if using a video is more convenient than having the computer in the flesh (in the silicon would be more correct). A pre-recorded video is in fact available from Philip Lawton.

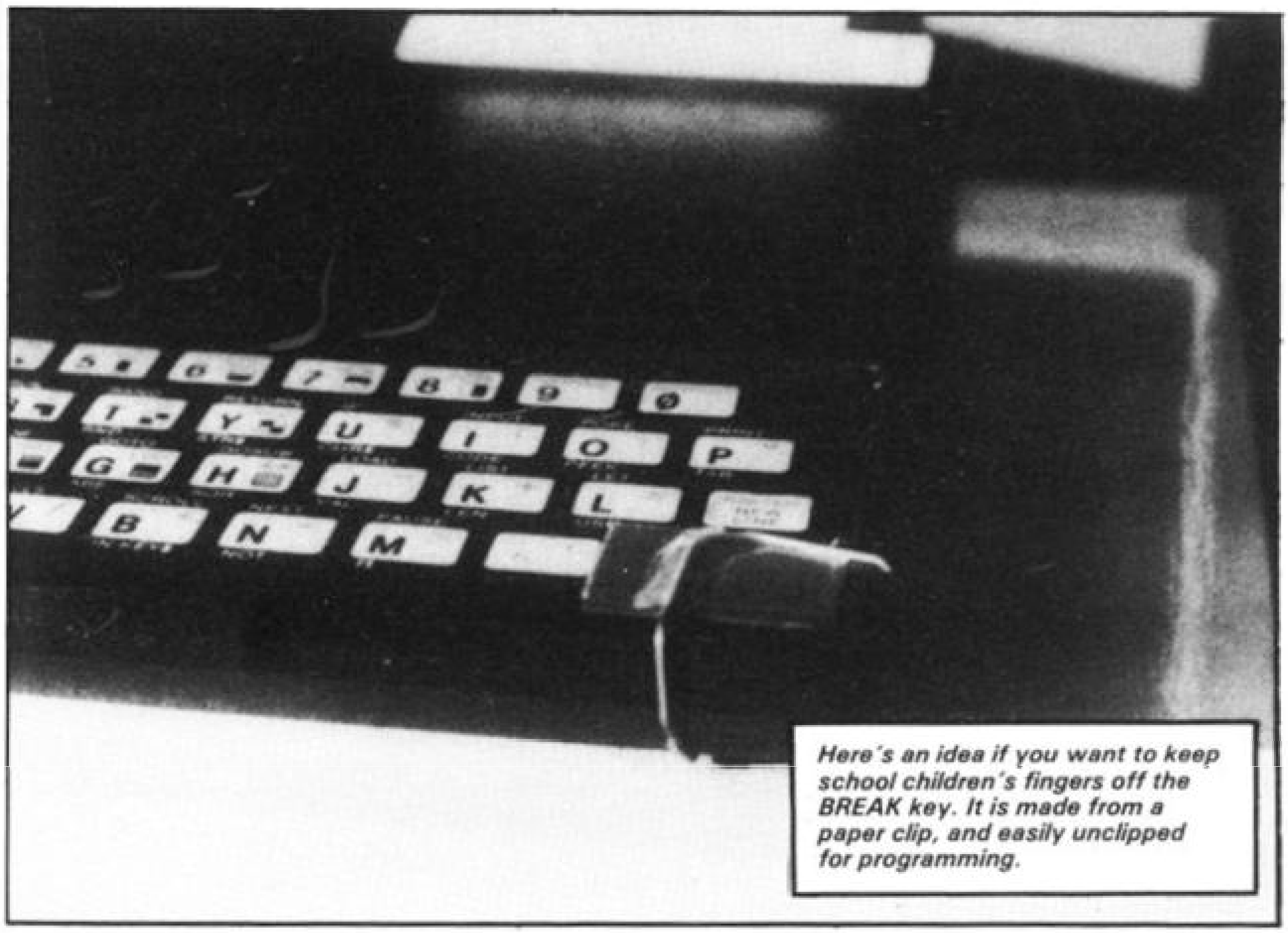
Looking now at the actual program, it is approximately 7½ K long and is supplied on a C12 cassette. It can be run in three different ways:

(i) Continuously going through each function.

(ii) Continuously going through each function but waiting for a response from the user.

(iii) Running particular areas of the programs, to demonstrate particular problems.





Here's an idea if you want to keep school children's fingers off the BREAK key. It is made from a paper clip, and easily unclipped for programming.

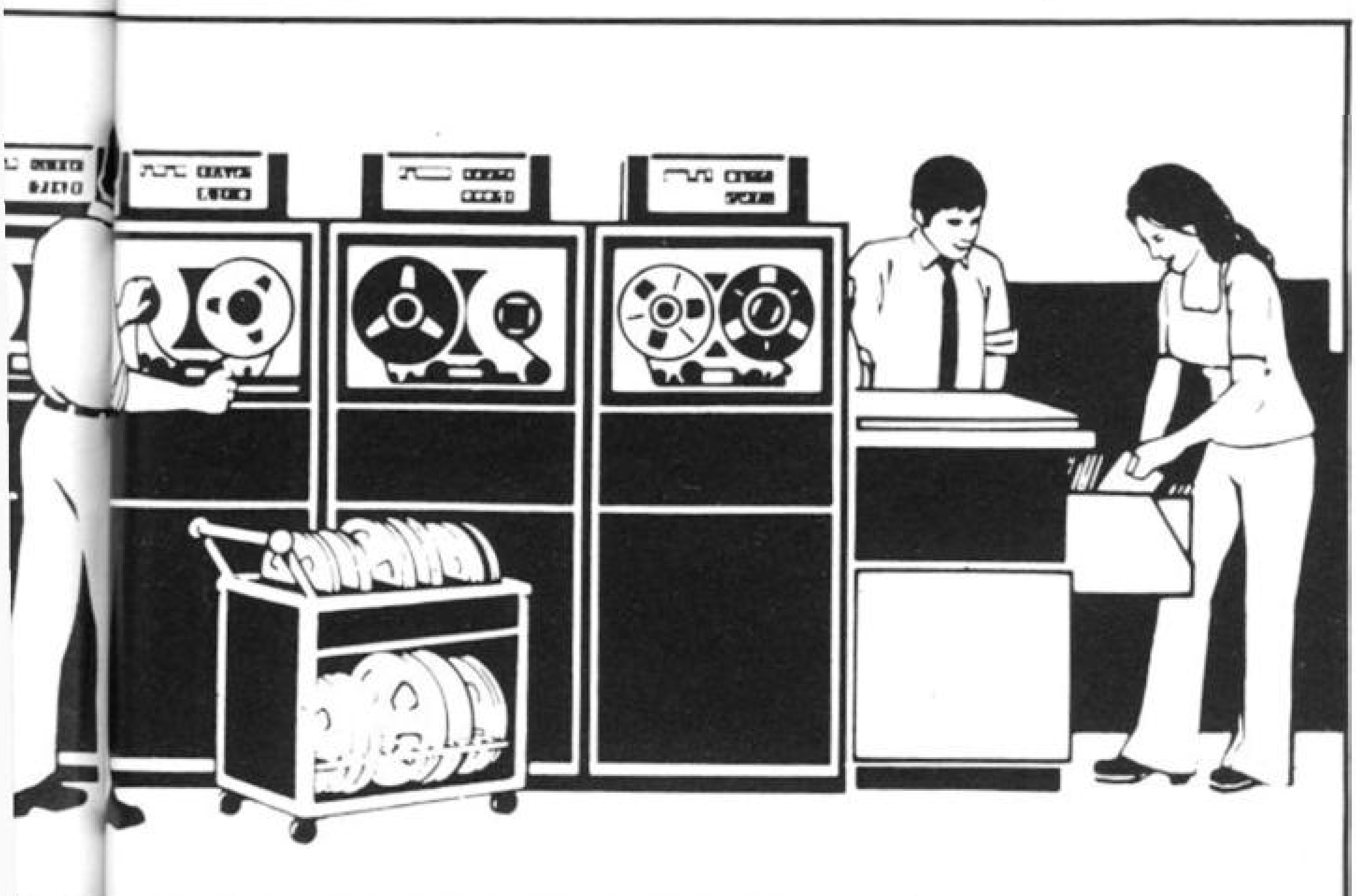
Conclusions

The whole thing has been very well thought out, and the idea of video recording a session with the package may appeal to many tutors far more than lugging a computer and all the various wires and modules around. Good provision for the student has also been made, with extra copies of graphs and written permission for copies to be made in the purchasers school or other similar educational establishment.

This is generally a good package and worth considering. Details of other titles are available from:
PHILIP LAWTON
4 TEMPLAR WAY
ROTHLEY
LEICESTER LE7 7LN.
please include a stamped addressed envelope.

Getting back to primaries

EdZX is a company specialising in educational software for the ZX computers.



The company is currently offering two programs: NUMPRAC and SPELL. SPELL is supplied with a supplementary program LETTERHUNT, and accessory Keyboard Overlay.

NUMPRAC is a suite of seven number practice games ranging from 'Count the Blocks' for 4-year-olds to 'Series' for upper Primary; and including Varied Format questions in response to recent recommendations. Originally designed as the program with which to introduce computers into a school, NUMPRAC explains signs like *, emphasises the use of NEWLINE and the gentle touch on the keyboard.

The program has full input validation; it features bold reward-word graphics and sophisticated teacher's control facilities. NUMPRAC does not attempt to 'teach', it works by positive reinforcement.

LETTERHUNT and SPELL come together on one cassette. They share a common fount of bold lower-case letters and a lower-case Keyboard Overlay. The typeface on the overlay (Eurostile Bold) was chosen specially to match as closely as possible the characteristics of the letters obtainable with ZX81 graphics. The Overlay incorporates a BREAK key mask.

LETTERHUNT develops character recognition and keyboard familiarity, and is good preparation for SPELL.

SPELL is a substantial program intended to develop the quick recognition of words and the ability to respell them. (It is obviously not a 'reading' program since it is not concerned with the sound or meaning of words.)

The vocabulary is grouped according to word type and is safe from RUN.

The vocabulary is really incidental to the program and words may be added or deleted singly or in groups. This powerful facility enables the teacher (or better still, the children) to extend the vocabulary week by week witH new words. The teacher can control exactly which groups are presented to the children or can leave an element of choice to them.

NUMPRAC is £3.45, and LETTERHUNT/SPELL (cassette, documentation and keyboard overlay) is £3.95. Overlays are 65p each (£4.00 for 10). EdZX Educational Software (mail order only), 16 Grasmere Road, Dronfield Woodhouse, Sheffield, S18 5PS.

ZX81 GAMES

from

J.K. GREYE SOFTWARE

THE NEW GENERATION SOFTWARE HOUSE

"Without question the finest machine code games available today"......... J.N. ROWLAND Product Manager for W.H. SMITH.

GAMESTAPE 1 for 1K __________ only £3.95
10 Games incl. ASTEROIDS, UFO, CODE, BOMBER, GUILLOTINE, KALEIDESCOPE, etc.
PROBABLY THE BEST VALUE 1k TAPE AVAILABLE.

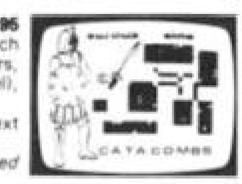


We've done in 1k games which some of our competitors required 16k to do!



GAMESTAPE 2 for 16K __________ only £3.95
***STARFIGHTER** Superb machine code Space Battle. Set against a background of twinkling stars, with stunning explosions – if you can hit the enemy!
PYRAMID Can you move the Pyramid? Make a mistake and it will colla....! A Thinkers game.
ARTIST The ultimate Graphic Designers aid. 8 Directions, 10 Memories, SAVE, COPY, RUBOUT, CLS, etc.

GAMESTAPE 3 for 16K __________ only £4.95
***CATACOMBS** A Multi-Level Graphics Adventure. Each level can contain up to 9 Rooms, 8 Passages, 7 Monsters, Food, Gold, Traps, Phantoms, an Exit (to the next level), and there's an infinite number of levels.
NOTE. . . . This is NOT one of the necessarily limited text Adventures as sold elsewhere.
"An excellent addictive game which will keep you amused for hours." . . . COMPUTER & VIDEO GAMES.





GAMESTAPE 4 for 16K __________ only £4.95
***3D MONSTER MAZE** The Game to Top All Others. Unbelievable Graphics! Can you find your way through the Maze? The EXIT is there somewhere, but then so is a T.REX, and its after YOU! All in 3D (the T.REX will actually run towards you in full perspective!), you've never seen anything like this before!
"3D MONSTER MAZE is the best game I have seen for the ZX81" . . . COMPUTER & VIDEO GAMES
"If I had to choose just one programme to impress an audience with the capabilities of the ZX81, then J.K. Greye's 3D MONSTER MAZE would be the one without doubt" . . . . ZX COMPUTING.

GAMESTAPE 5 for 16K __________ only £4.95
***3D DEFENDER** The Ultimate Space Game. Super fast Machine Code 3D version of the Arcade favourite. You have to save your home planet from the marauding Alien Spacecraft. This is all in 3D, your viewscreen shows you the view out of your fighters cockpit window. The backdrop moves when you turn, or fly up or down (8 flight directions), just as if you were really flying it! But then YOU ARE! The Enemy Saucers will actually zoom towards you in 3D, and shoot you if you let them! Your display includes Score, Shield Strength, Altitude, Proximity, Forward Radar and your viewscreen, which shows your rotating home planet, backdrop of Stars, Meteors, Explosions, Plasma Blasts, your Photon Beams, up to 4 Enemy Saucers and of course its all in full 3D!
A SMASH HIT at the ZX Microfair (most of the other software houses wanted a copy), a game not to be missed!





GAMESTAPE 6 for 1K __________ only £1.95
***BREAKOUT** Super Fast Full Screen Display Game. Your all time favourite with an added twist. See how much Money you can win and watch the pounds convert to Dollars. All in Machine Code for Fast Action with 3 Speeds, 2 Bat Sizes and three angles of rebound! The best BREAKOUT around and at this price you can't go wrong!

GAMES MARKED * INCL. MACHINE CODE.
Prices include VAT and U.K. P. & P.
(Add appropriate Postage on Foreign Orders). Cheques/P.O.s to

J.K. GREYE SOFTWARE

Dept.ZX, 16 Park St., Bath, Avon BA1 2TE.
CREDIT CARD SALES: Phone: 01-930-9232 (9 a.m. - 7 p.m.)
FOR INSTANT DESPATCH

If you prefer to see before buying, our range of **GAMESTAPES** are stocked by the following stores.

BUFFER MICROSHOP	374A, Streatham High Rd., London SW16
GEORGES	89 Park St., Bristol, Avon.
MICROSTYLE	29 Belvedere, Lansdown Rd., Bath, Avon.
MICROWARE	131 Melton Road, Leicester.
SCREEN SCENE	144 St. Georges Rd., Cheltenham, Glos.
W.H. SMITH	Computer Branches.
ZEDXTRA	5 School Lane, Kinson, Bournemouth, Dorset.

TRADE & EXPORT ENQUIRIES WELCOME

Now! For the SPECTRUM 48K!



FOOTBALL MANAGER

A GREAT GAME FROM

Addictive Games

for
ZX81
SPECTRUM
TRS80
VIDEO GENIE

JUST LOOK AT THESE FEATURES

★ *AS MANY SEASONS AS YOU LIKE* ★ *4 DIVISIONS* ★
★ *PROMOTION & RELEGATION* ★
★ *FULL LEAGUE TABLES* ★
★ *TRANSFER MARKET* ★
★ *INJURIES* ★
★ *7 LEVELS OF PLAY* ★
★ *SAVE GAME FACILITY* ★
★ *MANAGERIAL RATING* ★
★ *PICK YOUR TEAM FOR EACH MATCH* ★ *F.A. CUP* ★

This is NOT a mere simulation. This is an exciting and highly enjoyable game in which you play the part of a Football Manager coping with the problems and decisions involved in running your club. There are so many features it is impossible to list them here but included are form (top teams NORMALLY win), giant-killings, wage bills to pay, and you can even be sacked! It is a game requiring a great deal of skill, and people play it for literally hours on end (we have PROOF!).
WE GUARANTEE that this is one of the best computer games you've ever played!
BUT BEWARE, this game is extremely addictive!

HARDWARE REQUIRED

Spectrum 48K RAM	TRS80/ Video Genie LEVEL II 16K RAM
ZX 81 16K RAM	

To Order send Cheque/P.O. £7.95 made payable to: ADDICTIVE GAMES at: Dept. Z.X.C. P.O. BOX 278 *CONNIBURROW MILTON KEYNES MK14 7NE PLEASE STATE COMPUTER*

WHAT CAN I DO WITH 1K?

If your answer is "not much", then you must read Roger Valentine book, 'What Can I Do With 1K? (40 programs and routines for the 1K Sinclair ZX81)'.
". . .a splendid book and one which will repay your investment time and time again." (ZX Computing).

Book £4.95.
Also available on cassette £4.95.

NEW

WHAT CAN I DO WITH 16K?

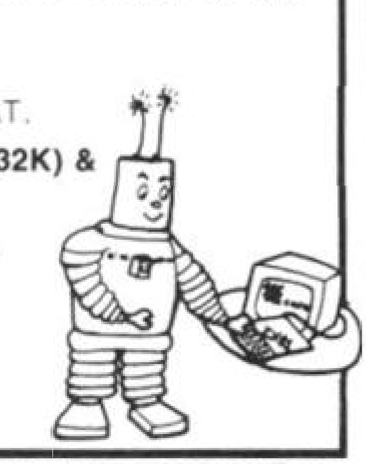
The companion volume to the above, containing complete program listings for 16K.

Book £4.95.
Cassettes available individually – ask for list.

ZX81 PAYROLL

Probably the best ZX program ever written for the serious business user.
Cassette (only) £12.65 inc VAT
Manual (only) £2.00
Cassette & Manual £14.50 inc VAT.

ALSO AVAILABLE FOR PET (32K) & SPECTRUM 48K



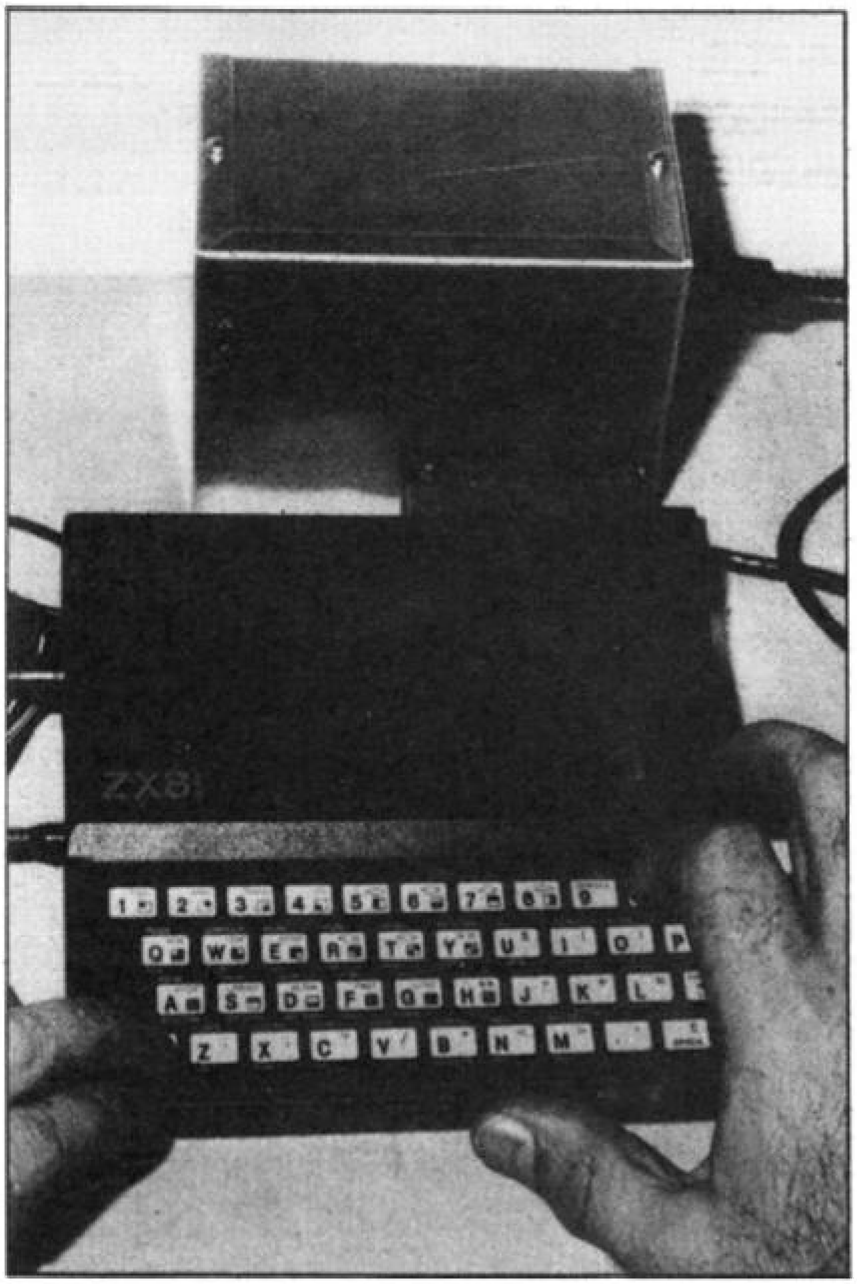
V&H Computer Services
182c Kingston Rd.
Staines
Middx.
Tel: Staines 58041

Watts that?

This program, written by D. Buckley of Aston-under-Lynne, is ideal for students studying physics, who have to carry out a number of Joule or Watt calculations.

When the program is run, a menu will appear, and 'joules' or 'watts' can be selected by entering the appropriate letter. All you have to do then is enter the figures, and up comes the answer.

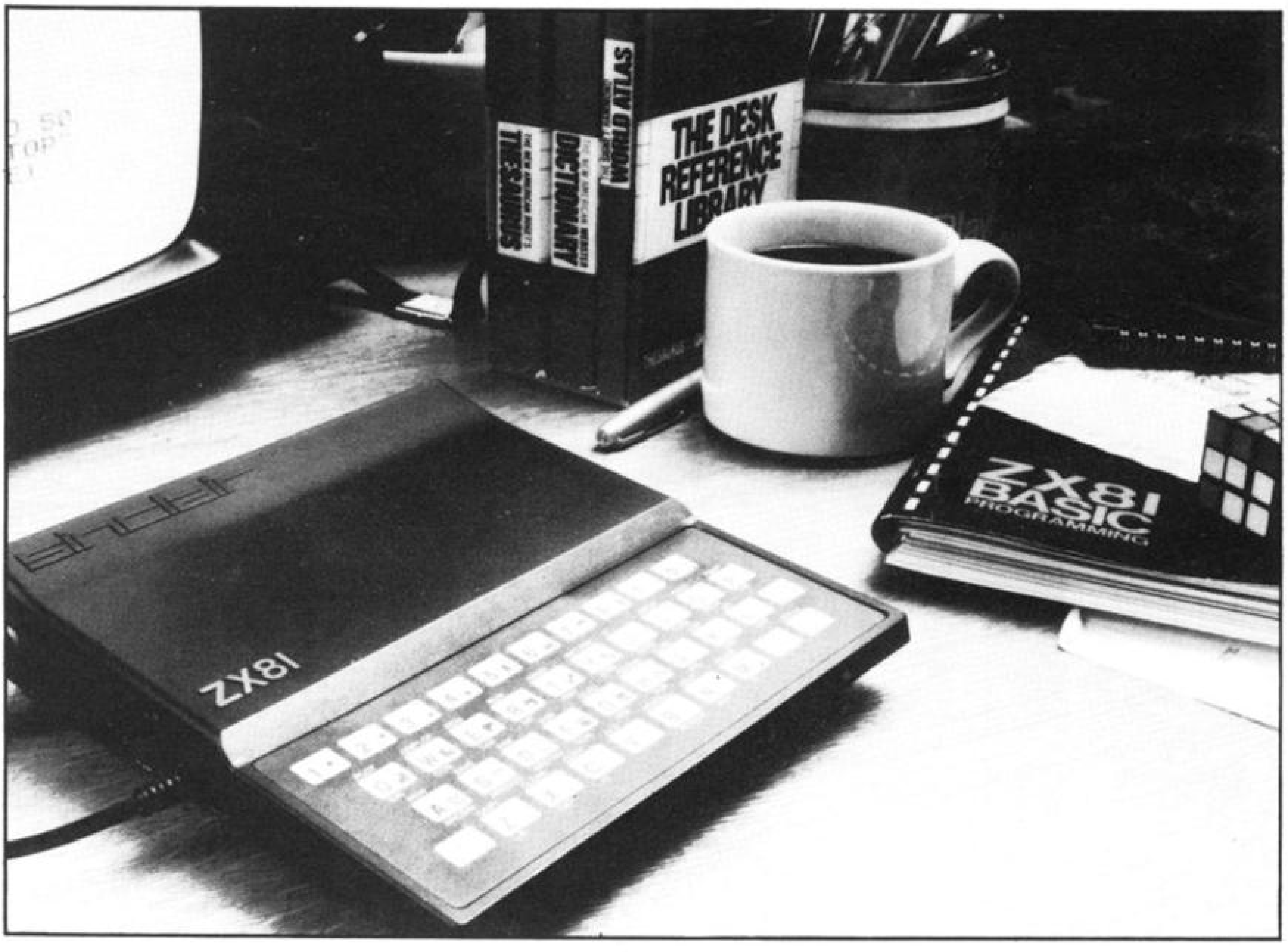
If you wish to use metres dragged against friction, instead of the height lifted, then simply enter the metres dragged when the program asks for height lifted.



```
 10 REM "JOULES"
 20 REM D.BUCKLEY
 25 CLS
 30 PRINT "JOULES/WATTS PROGRAM
",,,,
 40 PRINT "JOULES: INPUT A"
 50 PRINT "WATTS  : INPUT B"
 60 PRINT "STOP   : INPUT C"
 70 INPUT A$
 80 IF A$="A" THEN GOTO 1000
 90 IF A$="B" THEN GOTO 2000
 100 IF A$="C" THEN PRINT ,,"END
OF PROGRAM"
 110 STOP
1000 REM **JOULES**
1005 CLS
1010 PRINT "JOULES"
1020 PRINT
1030 PRINT "WEIGHT (NEWTONS)=?"
1040 INPUT A
1050 PRINT A
1060 PRINT
1070 PRINT "HEIGHT LIFTED(METRES
)=?"
1080 INPUT B
1090 PRINT B
1100 LET C=A*B
1110 PRINT AT 10,0;A;" NEWTONS *
";B;" METRES=";C;" JOULES"
1120 PRINT ,,,,
1130 PRINT "RETURN TO MENU: INPUT
A"
1140 PRINT "CONTINUE        : INPUT
B"
1150 INPUT A$
1160 IF A$="A" THEN GOTO 25
1170 IF A$="B" THEN GOTO 1000
1180 STOP
2000 REM **WATTS**
2005 CLS
2010 PRINT "WATTS"
2020 PRINT
2030 PRINT "WEIGHT(NEWTONS)=?"
2040 INPUT A
2050 PRINT A
2060 PRINT
2070 PRINT "HEIGHT LIFTED(METRES
)=?"
2080 INPUT B
2090 PRINT B
2100 PRINT
2110 PRINT "TIME(SECONDS)=?"
2120 INPUT C
2130 PRINT C
2140 LET D=(A*B)/C
2150 PRINT AT 15,0;A;" NEWTONS *
";B;" METRES / ";C;" SECS. =";D
;" WATTS"
2160 PRINT
2170 PRINT "RETURN TO MENU: INPUT
A"
2180 PRINT "CONTINUE        : INPUT
B"
2190 INPUT A$
2200 IF A$="A" THEN GOTO 25
2210 IF A$="B" THEN GOTO 2000
2220 STOP
```

ELECTRICAL CIRCUITS

From Paisley, Scotland, Thomas Ballantyne has sent us a program which was devised to calculate and illustrate — using circuit and phasor diagrams — the characteristics of a series AC electrical circuit. The program is for a 16K ZX81.



The program covers series AC circuits having: Resistance, and Inductance: Resistance and Capacitance: Resistance, Inductance and Capacitance. It has also been designed to cover the case of Series Resonance.

Calculations of: Reactance : Impedance : Current : Voltages : Power Factor : Power in Watts : Volt Amperes and Reactive Power are made and the results displayed. A circuit diagram is drawn. The circuit has the current displayed and the voltages across the components. Values are to the nearest whole number.

The program can then be used to illustrate the phasor diagram for the circuit. The diagram shows the in phase, and out of phase voltages, and the phase angle between current and applied voltage.

The user is prompted to enter the essential quantities one at a time. If capacitance is to be entered then it should be in farads (eg 200 Microfarads = 200E-6 farads). If there is no resistive and inductive only), then capacitance should be entered as 0. Even though theoretically this would give infinite reactance. The program is arranged to take account of this.

In similar fashion if there is no inductance enter this as 0. No provision has been made for zero resistance, this being an unlikely occurrence. However, if 0 is entered for resistance the program will run normally, until it reaches the stage of displaying the phasor diagram. The diagram will appear on the screen and a code 6 will indicate an arithmetic overflow.

The program may be run in *fast* or *slow*. *Slow* allows the diagrams to build up, and can be fascinating to watch. The program was originally devised to assist in the teaching of this subject to electrical students who were apprentices. A further adaptation is planned to make the student do a bit more work than the computer, the basis of real learning.

```
   1 REM "SERIES A.C.CIRCUIT"
   2 REM "T.BALLANTYNE 1982"
   3 PRINT "PROGRAM REQUIRES"
   4 PRINT "RESISTANCE IN OHMS"
   5 PRINT "INDUCTANCE IN HENRYS
"
   6 PRINT "CAPACITANCE IN FARADS
"
   7 PRINT "FREQUENCY IN HERTZ
      ENTER"
   8 PRINT "SUPPLY VOLTAGE IN VO
LTS   THE"
   9 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒   VALUES"
  10 PRINT "RESISTANCE R=";
  20 INPUT R
  30 PRINT R
  40 PRINT "INDUCTANCE L=";
  50 INPUT L
  60 PRINT L
  70 PRINT "CAPACITANCE C=";
  80 INPUT C
  90 PRINT C
 100 PRINT "FREQUENCY F=";
 110 INPUT F
 120 PRINT F
 130 PRINT "SUPPLY VOLTAGE E=";
 140 INPUT E
 150 PRINT E
 160 PAUSE 250
 170 POKE 16437,255
 180 CLS
 190 LET XL=2*PI*F*L
 200 IF C=0 THEN GOTO 630
 210 PRINT
 220 PRINT "INDUCTIVE REACTANCE
XL"
 225 PRINT "          =";XL;" OHMS"
 226 PRINT
 228 IF C=0 THEN GOTO 265
 230 LET XC=1/(2*PI*F*C)
 235 PRINT
 236 PRINT
 238 PRINT
 240 PRINT "CAPACITIVE REACTANCE
XC"
 245 PRINT "          =";XC;" OHMS"
 250 IF XL>XC THEN GOTO 650
 260 LET Z=SQR ((R*R)+(XC-XL)*(X
C-XL))
 265 PAUSE 200
 266 POKE 16437,255
 268 CLS
 270 PRINT
 280 PRINT "IMPEDANCE Z=";Z;" OH
MS"
 290 LET I=E/Z
 300 PRINT
 310 PRINT "CURRENT I"
 315 PRINT "          =";I;" A."
 320 LET VR=I*R
 330 PRINT
 340 PRINT "P.D.ACROSS RESISTOR
VR"
 345 PRINT "          =";VR;" V."
 350 LET VL=I*XL
 360 PRINT
 370 PRINT "P.D.ACROSS INDUCTOR
VL"
 375 PRINT "          =";VL;" V."
 380 IF C=0 THEN GOTO 430
 390 LET VC=I*XC
 400 PRINT
 410 PRINT "P.D.ACROSS CAPACITOR
VC"
 415 PRINT "          =";VC;" V."
 420 IF INT (XL+0.5)=INT (XC+0.5
) THEN GOTO 670
 430 LET PF=R/Z
 440 PRINT
 450 IF C=0 THEN GOTO 480
 455 IF XL>XC THEN GOTO 480
 460 PRINT "POWER.FACTOR=";PF;"
LEAD"
 470 GOTO 490
 480 PRINT "POWER FACTOR=";PF;"
 LAG"
 500 LET Y=ACS PF
 510 PRINT
 515 LET G=(Y*180)/PI
 520 PRINT "PHASE ANGLE=";G;" DE
GS"
 523 PRINT
 524 PRINT "PRESS CONT BUTTON AN
D NEWLINE"
 527 STOP
 530 LET W=E*I*PF
 540 PRINT
 550 PRINT "POWER=";W;" W."
 560 LET S=E*I
 570 PRINT
 580 PRINT "APPARENT POWER=";S;"
VA."
 590 LET Q=E*I*SIN Y
 600 PRINT
 610 PRINT "REACTIVE POWER=";Q;"
VAR."
 612 PRINT
 613 PRINT
 614 PRINT "PRESS D AND NEWLINE"
 615 PRINT "██████████████████"
 616 PRINT "FOR CIRCUIT DIAGRAM"
 617 PRINT "██████████████████"
 618 INPUT A$
 619 IF A$="D" THEN GOTO 730
 630 LET Z=SQR ((R*R)+(XL*XL))
 635 PRINT "CAPACITIVE REACTANCE
=0"
 640 GOTO 210
 650 LET Z=SQR ((R*R)+(XL-XC)*(X
L-XC))
 660 GOTO 265
 680 PRINT
 690 PRINT "CIRCUIT IS AT,OR"
 695 PRINT "NEAR RESONANCE"
 700 LET PF=1
 705 PRINT "POWER FACTOR =1"
 720 GOTO 500
 730 CLS
 740 IF C=0 THEN GOTO 1060
 750 IF XL=0 THEN GOTO 1290
 760 FOR X=0 TO 10
 770 PLOT X,38
 780 NEXT X
 790 FOR X=22 TO 28
 800 PLOT X,38
 810 NEXT X
 820 FOR X=38 TO 48
 830 PLOT X,38
 840 NEXT X
 850 PRINT AT 2,5;"▒▒▒▒▒▒"
 860 PRINT AT 2,14;"UUUUUU"
 865 PRINT AT 1,24;"█ █"
 870 PRINT AT 2,24;"█ █"
 880 PRINT AT 3,24;"█ █"
 890 FOR X=51 TO 63
 900 PLOT X,38
 910 NEXT X
 920 PRINT AT 1,8;"R=";R
 930 PRINT AT 1,16;"L=";L
 940 PRINT AT 0,24;"C=";C
 950 GOSUB 1500
 960 PRINT AT 5,5;INT (VR+0.5);"
V."
 970 PRINT AT 5,16;INT (VL+0.5);
" V."
 980 PRINT AT 5,24;INT (VC+0.5);
" V."
 985 PRINT AT 10,2;INT (I+0.5);"
A."
1010 GOSUB 2000
1060 FOR X=0 TO 10
1070 PLOT X,38
1080 NEXT X
1090 FOR X=22 TO 39
1100 PLOT X,38
1110 NEXT X
1120 FOR X=50 TO 63
```

```
1130 PLOT X,38
1140 NEXT X
1150 PRINT AT 2,5;"▒▒▒▒▒▒▒▒"
1160 PRINT AT 2,20;"UUUUUU"
1170 PRINT AT 1,8;"R=";R
1180 PRINT AT 1,22;"L=";L
1190 GOSUB 1500
1200 PRINT AT 4,5;INT (VR+0.5);"
V."
1210 PRINT AT 4,20;INT (VL+0.5);
" V."
1215 PRINT AT 10,2;INT (I+0.5);"
A."
1240 GOSUB 2000
1290 FOR X=0 TO 10
1300 PLOT X,38
1310 NEXT X
1320 FOR X=22 TO 43
1330 PLOT X,38
1340 NEXT X
1350 FOR X=48 TO 63
1360 PLOT X,38
1370 NEXT X
1380 PRINT AT 2,5;"▒▒▒▒▒▒▒▒"
1390 PRINT AT 1,22;"███"
1400 PRINT AT 2,22;" █ "
1405 PRINT AT 3,22;" █ "
1410 PRINT AT 0,8;"R=";R
1420 PRINT AT 0,23;"C=";C
1430 GOSUB 1500
1440 PRINT AT 5,3;INT (VR+0.5);"
V."
1450 PRINT AT 5,19;INT (VC+0.5);
" V."
1455 PRINT AT 10,2;INT (I+0.5);"
A."
1460 GOSUB 2000
1500 FOR Y=0 TO 38
1510 PLOT 0,Y
1515 NEXT Y
1517 FOR Y=0 TO 38
1520 PLOT 63,Y
1530 NEXT Y
1540 FOR X=0 TO 21
1550 PLOT X,0
1560 NEXT X
1570 FOR X=42 TO 63
1580 PLOT X,0
1590 NEXT X
1600 PRINT AT 21,15;"E"
1610 PRINT AT 19,15;E;" V."
1620 RETURN
2000 PRINT AT 12,2;"PRESS P THEN
 NEWLINE"
2010 PRINT AT 13,2;"FOR PHASOR D
IAGRAM"
2020 INPUT Z$
2030 IF Z$="P" THEN GOTO 4000
2040 RETURN
4000 CLS
4010 IF L=0 THEN GOTO 6000
4020 IF C=0 THEN GOTO 5000
4025 IF INT (XL+0.5)=INT (XC+0.5
) THEN GOTO 4050
4030 IF XC<XL THEN GOTO 7000
4040 IF XC>XL THEN GOTO 8000
4050 GOSUB 9000
4060 PRINT AT 10,24;"E=";E;" V."
4070 PRINT AT 14,18;"0=ZERO DEGR
EES"
4080 PRINT AT 15,18;"VR=E"
4090 PRINT AT 16,18;"RESONANCE"
4100 PRINT AT 17,18;"▒▒▒▒▒▒▒▒▒"
4110 STOP
5000 FOR X=0 TO 56
5010 PLOT X,4
5020 NEXT X
5030 FOR Y=4 TO 40
5040 PLOT 0,Y
5050 NEXT Y
5060 PRINT AT 2,2;"VL=";INT (VL+
0.5);" V."
5070 PRINT AT 18,24;"VR=";INT (V
R+0.5);" V."
5080 PRINT AT 20,24;"I=";INT (I+
0.5);" A."
5085 LET M=TAN (G*2*PI/360)
5090 FOR X=0 TO 20
5100 PLOT X,INT (M*X)+4
5110 NEXT X
5120 PRINT AT 6,20;"E=";E;" V."
5130 PRINT AT 18,6;"0"
5140 PRINT AT 4,4;"LAGGING P.F."
5145 PRINT AT 5,4;"ANGLE 0=";INT
 G;" DEGS"
5150 STOP
6000 FOR X=0 TO 56
6010 PLOT X,40
6020 NEXT X
6030 FOR Y=0 TO 40
6040 PLOT 0,Y
6050 NEXT Y
6060 PRINT AT 18,2;"VC=";INT (VC
+0.5);" V."
6070 PRINT AT 1,24;"VR=";INT (VR
+0.5);" V."
6080 PRINT AT 3,24;"I=";INT (I+0
.5);" A."
6085 LET M=TAN (G*2*PI/360)
6090 FOR X=1 TO 40
6095 IF -M*X+40<0 THEN GOTO 6120
6100 PLOT X,INT (-M*X)+40
6110 NEXT X
6120 PRINT AT 14,24;"E=";E;" V."
6130 PRINT AT 2,3;"0"
6140 PRINT AT 16,4;"LEADING P.F.
"
6145 PRINT AT 17,4;"ANGLE 0=";IN
T G;" DEGS"
6150 STOP
7000 GOSUB 9000
7005 PRINT AT 4,2;"(VL-VC)=";INT
 (VL-VC);" V."
7008 LET M=TAN (G*2*PI/360)
7010 FOR X=0 TO 20
7020 PLOT X,INT (M*X)+20
7030 NEXT X
7040 PRINT AT 10,6;"0"
7050 PRINT AT 2,24;"E=";E;" V."
7060 PRINT AT 18,20;"LAGGING P.F
."
7065 PRINT AT 19,16;"ANGLE 0=";I
NT G;" DEGS"
7070 STOP
8000 GOSUB 9000
8005 PRINT AT 18,2;"(VC-VL)=";IN
T (VC-VL);" V."
8008 LET M=TAN (G*2*PI/360)
8010 FOR X=0 TO 20
8015 IF -M*X+20<0 THEN GOTO 8040
8020 PLOT X,INT (-M*X)+20
8030 NEXT X
8040 PRINT AT 12,3;"0"
8050 PRINT AT 17,24;"E=";E;" V."
8060 PRINT AT 20,14;"LEADING P.F
."
8065 PRINT AT 21,14;"ANGLE 0=";I
NT G;" DEGS"
8070 STOP
9000 FOR X=0 TO 56
9010 PLOT X,20
9020 NEXT X
9030 PRINT AT 11,24;"I=";INT (I+
0.5);" A."
9040 FOR Y=20 TO 40
9050 PLOT 0,Y
9060 NEXT Y
9070 PRINT AT 2,2;"VL=";INT (VL+
0.5);" V."
9080 FOR Y=20 TO 0 STEP -1
9090 PLOT 0,Y
9100 NEXT Y
9110 PRINT AT 20,2;"VC=";INT (VC
+0.5);" V."
9120 PRINT AT 9,23;"VR=";INT (VR
+0.5);" V."
9130 RETURN
```

```
PROGRAM REQUIRES
RESISTANCE IN OHMS
INDUCTANCE IN HENRYS
"CAPACITIVE IN FARADS
FREQUENCY IN HERTZ
SUPPLY VOLTAGE IN VOLTS          ENTER
                                 THE
                                 VALUES
RESISTANCE R=10
INDUCTANCE L=.05
CAPACITANCE C=.0003
FREQUENCY F=50
SUPPLY VOLTAGE E=200

IMPEDANCE Z=11.224343 OHMS

CURRENT I
      =17.818415 A.

P.D.ACROSS RESISTOR VR
      =178.18415 V.

P.D.ACROSS INDUCTOR VL
      =279.89101 V.

P.D.ACROSS CAPACITOR VC
      =189.05926 V.

POWER FACTOR=0.89092077    LAG

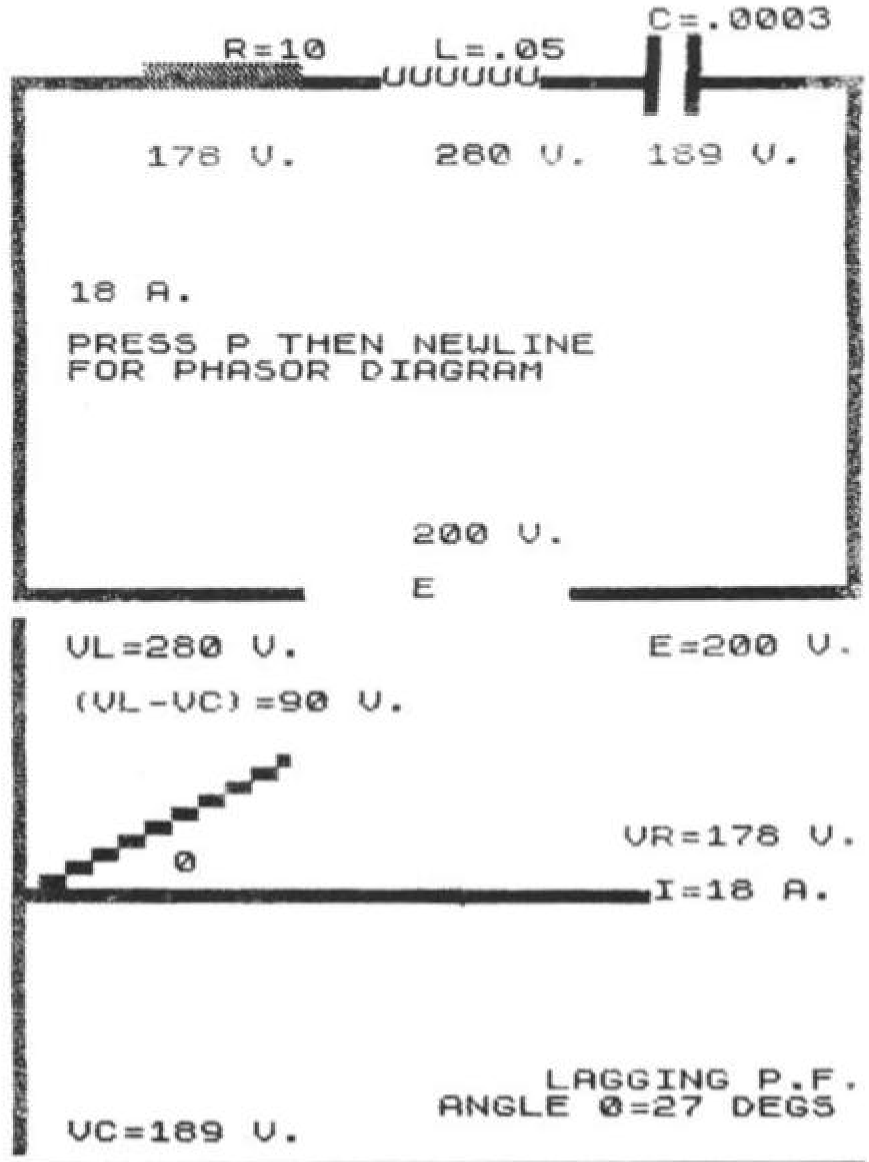
PHASE ANGLE=27.010821 DEGS

PRESS CONT BUTTON AND NEWLINE


POWER=3174.9592 W.

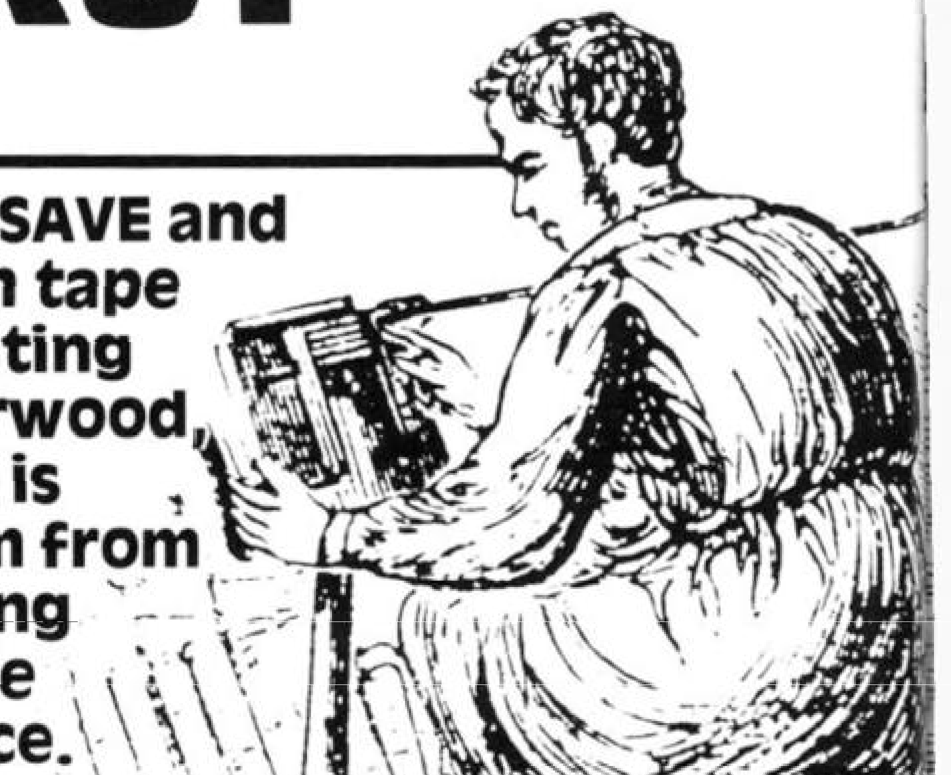
APPARENT POWER=3563.6831 VA.

REACTIVE POWER=1618.4779 VAR.
```



Preparing a tape directory for the ZX81

The ability of the ZX81 to SAVE and LOAD a named program on tape opens up new and interesting possibilities. James Calderwood, Coleraine, explains how it is possible to load a program from a directory, by just entering the number printed beside the program of your choice.



The obvious way to LOAD a program is to type LOAD "LUNAR" followed by NEWLINE. However, it is possible to use an expression such as LOAD E$ where E$ has the value "LUNAR" or whatever the required program name may be, eg
LET E$ = "LUNAR"
LOAD E$
has the same effect as LOAD "LUNAR". Not very useful, you may say, two lines to type instead of one; but as I will show in the program following, this fact can be used to good effect. There is just one other point to understand before we see how to develop a directury program. It may seem that if we can use the expression LOAD E$ we could equally use LOAD A$(5) or LOAD A$(C). Well, we could, except for one little problem. If A$(C) has been DIMentioned as, for example, A$(12,12) then A$(5) would not be LUNAR but LUNAR followed seven spaces to bring it up to 12 characters in length. This would not be recognised as the title given when the program was SAVEd as LUNAR.

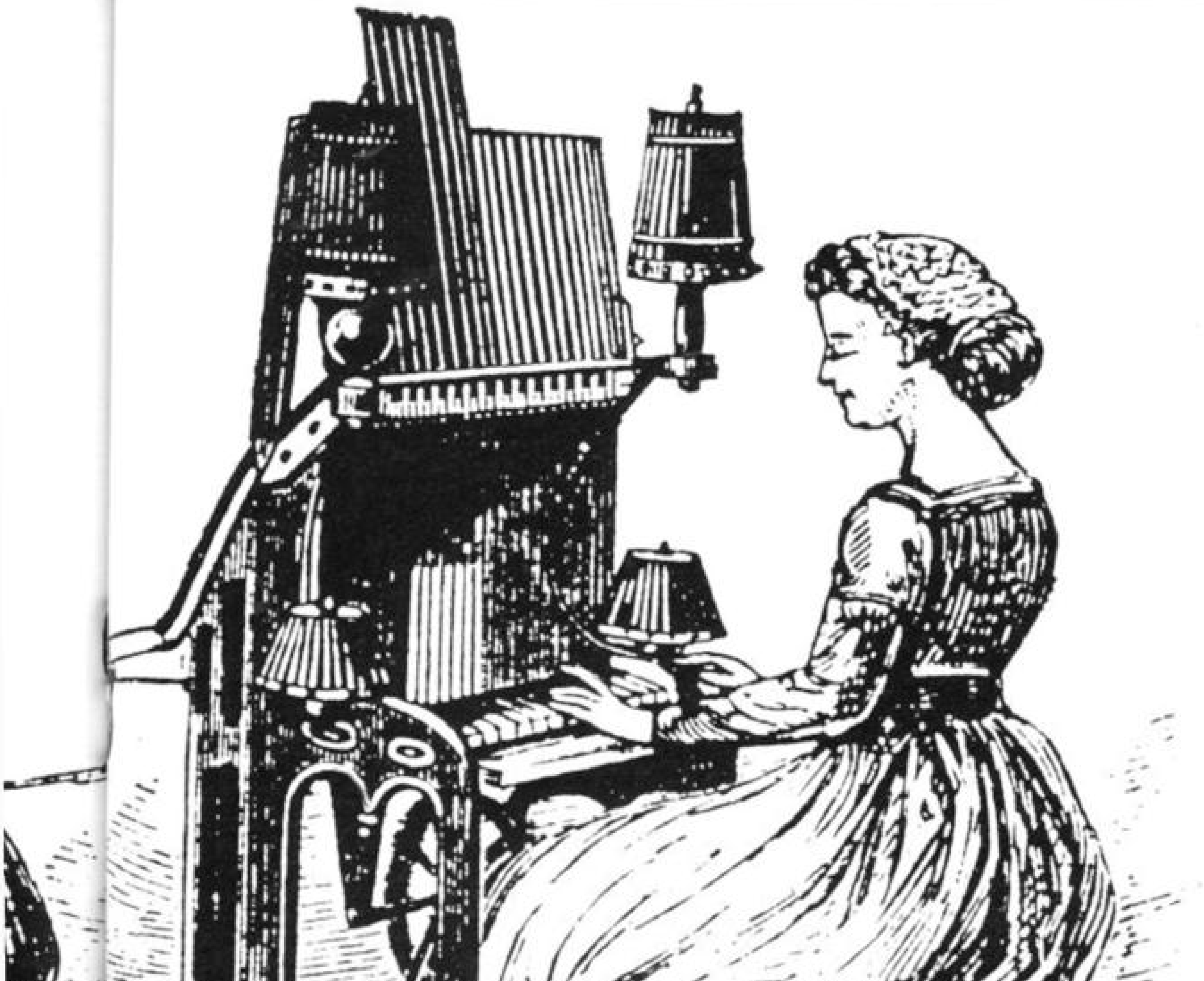
This problem is overcome by putting a full stop "." after each program title in the directory. Your ZX81 will recognise this as the end of the title and LOAD the required program. (You must not, of course, use the full stop in the title when you SAVE the program.)

PREPARING THE DIRECTORY

How you prepare your directory will depend on the amount of memory at your disposal. The program I have given here uses about 2K, so let us look at some factors influencing the memory needed, and see how we can adapt the directory to fit 1K if this is the memory available. Here are some economies we can make:-

1. The DIM statements in lines 10 and 20 are dependent on the number of programs to be recorded on the tape and could vary from 4 or 5 to 20 or 30 depending on the length of tape used. The second argument of B$ is the maximum length of the title to be used. So the shorter the title we are prepared to accept the less memory needed. I have found that ten titles of eight letters each (ie 7 + ".") is quite satisfactory and very economical.

2. Once your directory is complete you do not need lines 10 to



70, except for editing, and I will explain how this can be done in command mode. This means that you can first type in lines 10 to 70 and 300 to 320. When you RUN this you can enter as many programs as you have on your tape, complete with their starting position. Then by entering "0" the directory will be SAVEd on a spare piece of tape, in case you want this part again. Lines 10 to 70 can now be erased (just type 10 NEWLINE, etc), and the rest of the program typed in. You will have to omit any reference to these earlier lines, eg in line 120 omit "OR99 TO EDIT" and omit line 140 altogether. When you have finished typing the directory the command GOTO 300 will SAVE it at the start of your tape (you've left plenty of room, haven't you?).

Now when you type LOAD "DIRECTORY" or just LOAD "" the directory will load and continue from line 80, printing out a list of programs and asking "PROGRAM NUMBER?" When you respond with the number of your chosen program you will be told where to start your tape for your program to be loaded automatically. You may be wondering what to do if you want to add programs to your tape, after all how often do you start with a tape full of programs? This is where lines 25 to 70 would have been useful.

For those lucky enough to have 16K RAM packs here is what you do: On the prompt "PROGRAM NUMBER?, OR 99 to EDIT" you type 99. You will be asked for "PROGRAM NUMBER?", "PROGRAM NAME?", and "PROGRAM START". This will update your directory and be SAVEd when you reply "0" to "PROGRAM NUMBER". Without this EDIT facility we must work in command mode. To do this LOAD the directory and get into command mode. Type, eg

```
LET C=7
LET B$="LUNAR"
LETA(7)=65
GOTO 300
```

Your directory is now updated and recorded.

I find it useful to add one more line to my list of programs in the directory if there is any more room on the tape for additional programs. I enter at the next available number a line such as "PROGRAMS END AT" and then give the counter reading.

The Program

```
10 DIM A(12)
20 DIM B$(12,12)
25 CLS
27 PRINT "PROGRAM NUMBER?"
30 INPUT C
31 IF C=0 THEN GOTO 300
35 PRINT "PROGRAM NAME?"
40 INPUT B$(C)
45 PRINT "PROGRAM STARTS?"
50 INPUT A(C)
70 GOTO 25
80 FOR C=1 TO 12
90 IF A(C)=0 THEN GOTO 110
95 PRINT C;TAB4;B$(C);TAB 20;A(C)
100 NEXT C
110 PRINT
120 PRINT "PROGRAM NUMBER? OR 99 to EDIT"
130 INPUT D
140 IF D=99 THEN GOTO 25
150 FOR C=1 TO 12
160 IF B$(D,C)="." THEN GOTO 180
170 NEXT C
180 PRINT
190 PRINT "START TAPE AT ";A(D);" THEN NEWLINE"
200 INPUT C$
210 LOAD B$(D,1 TO C-1)
300 PRINT "START TAPE THEN PRESS NEWLINE"
301 PRINT "TO RECORD THE DIRECTORY"
310 INPUT C$
320 SAVE "DIRECTORY"
330 GOTO 80
```

Linear programming, and OPTIMAX

Linear programming is a mathematical technique that has been used to solve all sorts of problems, by performing an optimisation on information you supply it. Hiderbay's program OPTIMAX is designed to give the ZX81 owner access to this powerful decision-aiding technique.

With linear programming, you tell it how to measure your objective, and what the restrictions are, and it works out, firstly if it is feasible or not, and if it is, the best value of the result you can expect, and how to go about getting that result.

The Hilderbay program, OPTIMAX, is written in machine code, which not only enables a sophisticated program to fit into a small space, but makes the program very fast.

The program is supplied with a very detailed manual, which has been prepared with two objectives in mind; firstly to define how to use OPTIMAX, and secondly, to give somebody with no knowledge of linear programming a guide to how to set about formulating a problem. For those who know something about linear programming already, a section on how to apply that knowledge to OPTIMAX is included, so that they can make use of the progam as quickly as possible.

There are many problems, in business and elsewhere, which do not have a unique solution. Thus rather than asking 'What is the solution to the problem?', we are asking 'What is the best solution to the problem?'. Finding this best solution is usually called optimisation. But what do we mean by 'best'?

Talking about best implies that we have an objective in mind. For any form of optimisation we must be able to express this objective in a mathematical form. Normally we are trying to maximise something, for example, profit; or minimise something, for example, energy needs.

Using terms like optimisation implies that there must be constraining influences on the solution. If there are no constraints there would not be a problem. Again, these constraints must be capable of being expressed in a mathematical form.

The first stage in an optimisation problem is thus to express the problem mathematically. Normally this stage takes the longest time as we have to obtain a large amount of information.

Linear programming is just one method of optimisation. For it to be used, all relationships must be linear. In other words, as far as each equation goes, each element must be independent of all other elements:

The users' guide with OPTIMAX has been written with a great deal of care, to aid the user in understanding how the program can be used in general, and how it can be used to help with a specific problem.

For example, here's a quote from the manual, which appears under the heading 'Formulating a problem for OPTIMAX':

"The first thing to realise is that all you are doing is creating a model of the problem. There is a direct correspondence between the real life situation and this model. For example, if you are mixing cattle food and there is a requirement for a minimum amount of protein to be included, then there will be a constraint in the model covering minimum protein content. Similarly, the objective of the model corresponds to your objective in real life.

"You must tell the model how to measure the objective, just as in the real life situation you must have some way of measuring your objective. For example, if you run a factory and you wish to maximise your profit, you might measure your total profit in terms of the number of each product you sell multiplied by the profit you make on each one. The equivalent of this method of measurement must be set up in the model."

Full details on OPTIMAX, and the other business programs available from Hilderbay can be obtained by ringing Mike Salem on 01-485 1059, or writing to 8/10 Parkway, Regents Park, London NW1 7AA (Telex 22870).

Sample output from OPTIMAX:

```
FUNCTION?
T
COMPLETE-VAL=-1666.6
NO.IN OPT   SHADOW-COST   DEC/INC
CONST 01-MIN-N VAL: 200.00      <
            3.3333        249.99
                          125.00
CONST 02-MIN-P VAL: 50.000      <
            20.000        83.333
                          27.776
END
FUNCTION?
```

Here is a summary of functions available, and their codes, in OPTIMAX:

- A — Create variable
- B — Amend variable
- C — Delete variable
- D — Create constraint
- E — Amend contsraint
- F — Delete constraint
- G — Create usage
- H — Amend usage
- I — Delete usage
- J — List all variables
- K — List all constraints
- L — List all usage by variable
- M — List all usages by constraint
- O — Optimise
- P — Copy screen to printer
- S — Display variables in solution
- T — Display constraints in solution
- W — Erase all data prior to entering new problem
- X — Save data
- Z — Load data

Tracking down those pounds

Not only can you play games in your ZX81, but now, with a number of programs, you can keep track of your personal finances.

Among these programs is the J P Gibbons Personal Banking System, which can also be used to check bank statements.

The system can be used to help in budgeting and is designed to be easy to use, even by relatively inexperienced computer users. A number of controls and facilities have been built in with this in mind.

The Personal Banking System includes a full page detailed bank account, which can be dumped to the printer, as well as automatic generation of standing orders on due dates, and validation of all entries.

You can correct any item previously entered (with single or multiple field correction), and enter a previously omitted item in the correct date order of the account.

You can search for any item or items by cheque number, description or amount. There is a continuous display of statement extract, which is updated continually during input of entry. A file of standing order details can be displayed, printed, added to, cancelled and amended.

The program is provided with a detailed user manual, and Mr Gibbons says he'll provide after sales maintenance. The Personal Banking System (including a cassette and users' manual) is £9.95. You can get a copy of the manual for £1.00. A 32K version, offering multiple accounts, more entries and bar charts — among other features — is also available. J P Gibbons, 14 Avalon Road, Orpington, Kent, BR6 9AX.

Database

Most businesses and many other activities require the filing of names and addresses for mailing and reference, and DATABASE — available from Campbell Systems — fulfills this function admirably.

Almost any kind of list can be accommodated, but most users just make use of name, address, interest codes, and text — as the standard data to be stored in each record of the file.

The file is maintained by machine code in a way that ZX81 BASIC cannot, such that no space is reserved until after it is filled with data.

All items and records are variable length, so there is no waste of space.

You start with 12688 bytes of file space, and extra RAM above the 32K address will be usable as well. There are various display formats and search methods.

All file handling, searching and display is done in machine code, so DATABASE is very fast. The program is designed to be crash-proof.

The cassette, which costs £10.00, is accompanied by a detailed user guide, and is available from Campbell Systems, 15 Rous Rd, Buckhurst Hill, Essex, IG9 6BL.

Business games

A company called CCS have developed two business games for the ZX81. They are AIRLINE and AUTOCHEF, which are available for £4.75 each.

In AIRLINE your objective is to make enough capital by trading at a profit to take-over British Airways. You are required to decide on the number of aircraft to operate, whether to buy or charter, the level of staffing and maintenance, whether to enter into long-term contracts for supply of fuel and whether to repay loans. Problems encountered are tax demands, strikes, cancelled flights, hi-jacks and aircraft crashes.

In AUTOCHEF your object is also to trade profitably, so that the company accumulates sufficient capital ($25 million) to take-over Trust House Forte in the shortest possible time.

The quality of the decision making is of even greater importance in achieving the objective than it is in Airline. From information supplied you have to decide on which type of outlet to operate, the price of the menu, whether to enter into loan contracts or purchase consignments of food or wines and the level of advertising, wages and dividends. You are warned that if results and dividends are insufficient to satisfy the shareholders, you will be made to resign. There are three levels of difficulty.

Cases Computer Simulations are at 14 Langton Way, London, SE3 7TL.

Video-Plan

Video-Plan, developed by Video Software Limited, is designed to enable the ZX81 to be used as an analysis tool. It performs many functions which could otherwise be carried out using an analysis book and calculator.

Video-Plan can be used for such tasks as keeping stock records, analysing sales orders or invoices, analysing expenditure by nominal headings, cash flow forecasting or production scheduling.

The heart of the system is a user-defined chart stored in the computer's memory (say 50 lines by 20 columns). Data may be added to the chart and a full range of calculations performed across the lines, together with column totalling and subtotalling. The TV screen acts as a window through which the chart may be viewed. This window, of course, can be moved under user control to any part of the chart.

The cassette is well designed, with two copies of the program on one side, and a spoken explanation of the Video-Plan demonstration supplied is on side two. The program is very long (14K?) and starts running automatically. It takes nearly eight minutes to load. No loading difficulties were experienced.

When you first get the program up on the screen, a menu of seven choices is displayed:

1 — set up new chart
2 — define functions/titles
3 — enter data
4 — move window
5 — save the system
6 — reset chart
7 — re-calculate totals

If you wish to set up a new chart, you are invited to enter the dimensions (line and columns). A chart which is too big is rejected by the program.

The program is supplied with a clearly written, detailed 20 page booklet. Although it may seem overwhelming at first, the demonstration application, along with spoken word description, should enable any careful user to discover the value of the program. In summary, this is a carefully written, well-documented program, which should prove a definite asset to a company wishing to carry out any of the tasks mentioned at the start of this article. It is available from Video Software Ltd., Stone Lane, Kinver, Stourbridge, West Midlands, DY7 6EQ.

Dratted VAT

In the last issue we had a couple of programs to help you work out VAT. Reader John Jameson says he can do the same things those programs did, but more simply. Here's his program:

```
VAT RATE IS 15 PER CENT

COST      VAT       COST+VAT
12.36     1.85      14.21
24.72     3.7       28.42
37.08     5.56      42.64
49.44     7.41      56.85
61.8      9.27      71.07
74.16     11.12     85.28
86.52     12.97     99.49
98.88     14.83     113.71
```

```
111.24      16.68       127.92
123.6       18.54       142.14
  10 REM CALCULATING VAT
  15 REM (C) J. JAMESON
  20 PRINT "IF VAT RATE IS 15 PE
R CENT"
  30 PRINT "JUST PRESS NEWLINE"
  40 PRINT "IF YOU WANT ANOTHER
RATE"
  50 PRINT "THEN ENTER THAT BEFO
RE"
  60 PRINT "PRESSING NEWLINE"
  70 INPUT A$
  80 IF A$="" THEN LET R=15
  90 IF A$<>"" THEN LET R=VAL A$
 100 SCROLL
 110 PRINT "THE VAT RATE IS ";R;
" PER CENT"
 120 SCROLL
 130 PRINT "HOW MUCH IS ONE ITEM
?"
 135 INPUT C
 136 CLS
 140 SCROLL
 141 PRINT "VAT RATE IS ";R;" PE
R CENT"
 142 SCROLL
 143 SCROLL
 145 PRINT "COST";TAB 10;"VAT";T
AB 20;"COST+VAT"
 150 FOR G=1 TO 10
 160 SCROLL
 170 PRINT G*C;TAB 10;INT (C*G*R
)/100;TAB 20;G*C+INT (R*C*G)/100
 180 NEXT G
```

Getting installed

This utility program will work out instalment payments, when no interest is charged.

The program prompts are self-explanatory.

```
  10 REM INSTALMENT PAYMENTS
  15 REM NO INTEREST CHARGED
  20 PRINT "ENTER CASH PRICE"
  30 INPUT P
  40 PRINT "ENTER DEPOSIT AS A P
ERCENTAGE","OF CASH PRICE"
  50 INPUT D
  60 LET D=D/100*P
  70 PRINT "OVER HOW MANY MONTHS
?"
  80 INPUT N
  90 LET M=(P-D)/N
 100 PRINT "THE MONTHLY PAYMENT"
 110 PRINT "WILL BE £";INT (100*
(M+.005))/100
```

Pints to kilometres

Simple metric conversions are handled by this program, provided by John Knight, of Cheshire.

```
  10 REM METRIC CONVERSIONS
  20 PRINT "ENTER YOUR CHOICE:"
  30 PRINT "1 - KILOS TO POUNDS"
  40 PRINT "2 - POUNDS TO KILOS"
  50 PRINT "3 - POUNDS TO GRAMS"
  60 PRINT "4 - GRAMS TO POUNDS"
  65 PRINT "5 - TO STOP"
  70 IF INKEY$="" THEN GOTO 70
  72 LET A=CODE INKEY$-28
  75 IF A=5 THEN STOP
  80 CLS
  90 IF A=1 THEN GOTO 500
 100 IF A=2 THEN GOTO 1000
 110 IF A=3 THEN GOTO 1500
 120 IF A=4 THEN GOTO 2000
 130 RUN
 490 REM ****************
 500 REM KILOS TO POUNDS
 510 PRINT "HOW MANY KILOS?"
 520 INPUT K
 530 PRINT K;" KILOGRAMS IS ";K*
2.2;" POUNDS"
 540 GOTO 3000
 990 REM ****************
1000 REM POUNDS TO KILOS
1010 PRINT "HOW MANY POUNDS?"
1020 INPUT P
1030 PRINT P;" POUNDS IS ";P/2.2
;" KILOGRAMS"
1040 GOTO 3000
1490 REM ****************
1500 REM POUNDS TO GRAMS
1510 PRINT "HOW MANY POUNDS?"
1520 INPUT P
1530 PRINT P;" POUNDS IS ";INT (
45360*P)/100;" GRAMS"
1540 GOTO 3000
1990 REM ****************
2000 REM GRAMS TO POUNDS
2010 PRINT "HOW MANY GRAMS?"
2020 INPUT G
2030 PRINT G;" GRAMS IS ";INT (G
/4.536)/100," POUNDS"
2990 REM ****************
3000 FOR G=1 TO 100
3010 NEXT G
3020 CLS
3030 RUN
```

Keeping covered

Martin Kempler sent us this program, to work out how much carpet you need to cover the floors of your home. As you can see from the sample run (for three rooms), the prompts are simple (even if they do assume your rooms are rectangular), and the output is easy to understand.

```
HOW MANY ROOMS?
WHAT IS LENGTH OF ROOM 1?
WHAT IS WIDTH OF ROOM 1?
AREA OF ROOM 1 IS 180
AND THE TOTAL AREA SO FAR
                    IS 180

WHAT IS LENGTH OF ROOM 2?
WHAT IS WIDTH OF ROOM 2?
AREA OF ROOM 2 IS 120
AND THE TOTAL AREA SO FAR
                    IS 300

WHAT IS LENGTH OF ROOM 3?
WHAT IS WIDTH OF ROOM 3?
AREA OF ROOM 3 IS 104
AND THE TOTAL AREA
                    IS 404
```

```
  10 REM AREA CALCULATOR
  15 LET TOTAL=0
  17 SCROLL
  20 PRINT "HOW MANY ROOMS?"
  30 INPUT R
  40 FOR Q=1 TO R
  50 SCROLL
  60 PRINT "WHAT IS LENGTH OF RO
OM ";Q;"?"
  70 INPUT L
  75 SCROLL
  80 PRINT "WHAT IS WIDTH OF ROO
M ";Q;"?"
  90 INPUT W
 100 LET A=W*L
 110 SCROLL
 120 PRINT "AREA OF ROOM ";Q;" I
S ";A
 130 LET TOTAL=TOTAL+A
 140 SCROLL
 150 PRINT "AND THE TOTAL AREA "

 160 IF Q<R THEN PRINT "SO FAR "
 180 SCROLL
 190 PRINT ,"IS ";TOTAL
 200 SCROLL
 210 SCROLL
 220 NEXT Q
```

Along the wire

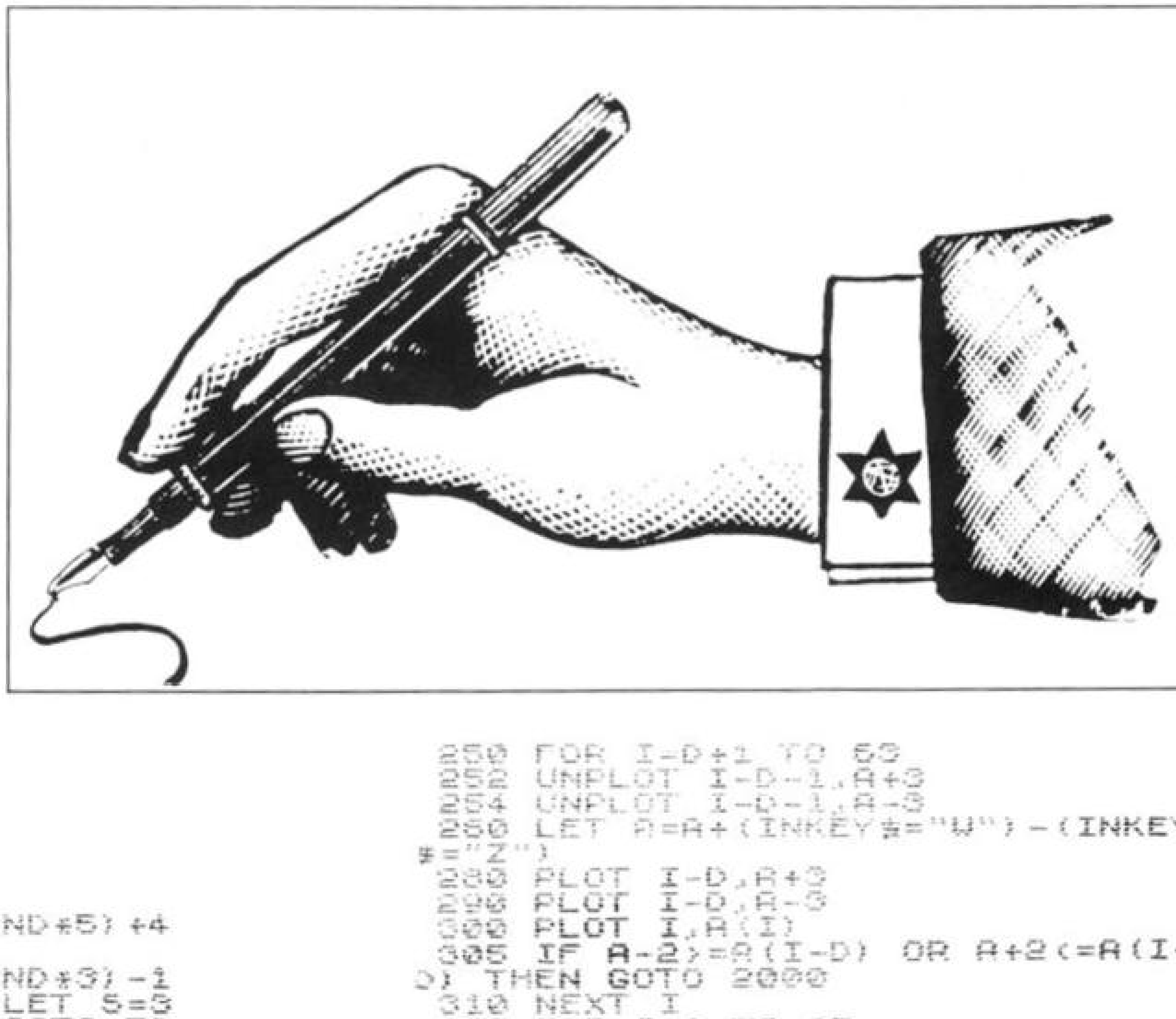
From Aughton in Lancashire, Daniel Haywood presents two challenging 16K programs — ALONG THE WIRE and a sophisticated SKETCH PAD — which allows you to save pictures you've created on tape.

The first program is ALONG THE WIRE. You have to guide a 'hoop' through a length of wire without touching the wire. You move at a constant speed, and can only see a small part of the wire in front of you.

You move with the "W" key (up) and the "Z" key (down). When you press NEWLINE or ENTER, you'll see the wire drawn out and you may study it for a short time . . . and then you are off.

Because some of the inverse print statements (in common with inverse print statements on ZX printers in general) are not too easy to read, we'll list them here for you:

330 PRINT AT 5,10;"WELL DONE". . .
1050 PRINT ,,,, TAB 5: "PRESS NEWLINE NOW"
2070 PRINT AT Y,X;"BUZZ"



```
   1 LET D=5
   2 GOTO 500
  10 CLS
  11 DIM A(63)
  20 LET Y=20
  30 LET A=1
  40 LET Q=0
  50 LET S=INT (RND*5)+4
  60 LET T=0
  70 LET R=INT (RND*3)-1
  80 IF R=0 THEN LET S=3
  90 IF Q=R THEN GOTO 70
 100 LET Q=R
 110 LET Y=Y+R
 120 IF Y>40 THEN LET R=-1
 130 IF Y<5 THEN LET R=1
 140 LET T=T+1
 150 LET A(A)=Y
 160 PLOT A,A(A)
 170 LET A=A+1
 180 IF A=64 THEN GOTO 200
 190 GOTO 50+(60 AND T<S)
 200 LET A=A(1)
 205 PRINT AT 0,0;"THAT IS THE C
OURSE","FOR YOU TO COMPLETE"
 210 FOR I=1 TO 100
 220 NEXT I
 225 CLS
 230 FOR I=1 TO D
 235 PLOT I,A(I)
 240 NEXT I
 250 FOR I=D+1 TO 63
 252 UNPLOT I-D-1,A+3
 254 UNPLOT I-D-1,A-3
 260 LET A=A+(INKEY$="W")-(INKEY
$="Z")
 280 PLOT I-D,A+3
 290 PLOT I-D,A-3
 300 PLOT I,A(I)
 305 IF A-2>=A(I-D) OR A+2<=A(I-
D) THEN GOTO 2000
 310 NEXT I
 320 FOR I=1 TO 25
 325 NEXT I
 327 CLS
 330 PRINT AT 5,10;"WELL DONE";A
T 12,3;"NOW TRY SOMETHING HARDER
"
 340 LET D=D-1
 350 FOR I=1 TO 100
 360 NEXT I
 370 CLS
 380 GOTO 10
 499 REM ***HIGH SCORE***
 500 LET HIGH=0
 510 LET H$=""
 999 REM ***INSTRUCTIONS***
1000 CLS
1005 PRINT TAB 13;"WIRE"
1010 PRINT AT 2,3;"YOU GUIDE A "
"HOOP""" THROUGH A","LENGTH OF WI
RE WITHOUT TOUCHING"
```

```
1020 PRINT "THE WIRE.YOU MOVE AT
 A CONSTANT  SPEED,AND CAN ONLY
SEE A SMALL PART OF THE WIRE IN
FRONT OF YOU"
1030 PRINT ,,"YOU MOVE WITH KEYS
,-",,"W-UP",,"Z-DOWN"
1040 PRINT ,,"WHEN YOU PRESS NEW
LINE,YOU WILL SEE THE WIRE DRAWN
 OUT AND YOU  MAY STUDY IT FOR A
 SHORT TIME";TAB 5;"AND THEN YOU
 ARE OFF"
1050 PRINT ,,,,TAB 5;"PRESS NEW
LINE NOW"
1055 IF INKEY$<>"" THEN GOTO 105
5
1060 IF CODE INKEY$<>118 THEN GO
TO 1060
1070 CLS
1080 GOTO 10
2000 LET Y=INT ((43-A)/2)
2010 LET X=(I-D)/2-2
2020 IF X<0 THEN LET X=0
2030 FOR Z=1 TO 10
2040 PRINT AT Y,X;"BUZZ"
2050 FOR N=1 TO 5
2060 NEXT N
2070 PRINT AT Y,X;"    "
2080 FOR N=1 TO 4
2090 NEXT N
2100 NEXT Z
2110 CLS
2120 LET SCORE=(5-D)*60+I-D
2121 PRINT AT 4,9;"YOUR SCORE IS
 ";SCORE
2122 IF HIGH>=SCORE THEN GOTO 21
34
2123 LET H$=""
2124 LET HIGH=SCORE
2125 PRINT AT 5,0;"YOUR SCORE IS
 THE HIGHEST-PLEASEENTER YOUR NA
ME (MAX. 6 LETTERS)";TAB 10;"- -
 - - - -"
2126 FOR A=1 TO 6
2127 IF INKEY$<>"" THEN GOTO 212
7
2128 LET A$=INKEY$
2129 IF A$="" THEN GOTO 2128
2130 LET H$=H$+A$
2131 PRINT AT 7,A*2+8;A$
2132 NEXT A
2133 GOTO 2135
2134 PRINT AT 7,0;" HIGH SCORE I
S ";HIGH;" HELD BY ";H$
2135 PRINT AT 10,2;"DO YOU WANT
ANOTHER GO (Y/N)?"
2136 GOTO 2136+(7844 AND INKEY$=
"N")+(4 AND INKEY$="Y")
2140 PRINT AT 12,1;"DO YOU WANT
INSTRUCTIONS (Y/N)?"
2145 IF INKEY$="" THEN GOTO 2145
2150 LET D=5
2160 GOTO 2150-(2140 AND INKEY$=
"N")-(1150 AND INKEY$="Y")
9980 CLS
9990 PRINT AT 11,13;"BYEEE"
9999 STOP
```

Sketch Pad

Unleash your artistic frustrations with this amazing program.

With SKETCHPAD, you can draw on the screen, clear the screen, save the picture on tape or transfer the screen to printer.

The keys to move the flashing cursor are shown when you run the program (instructions start at line 5010). You can also change the mode of drawing (ie a line or no line) by pressing "9" and "0".

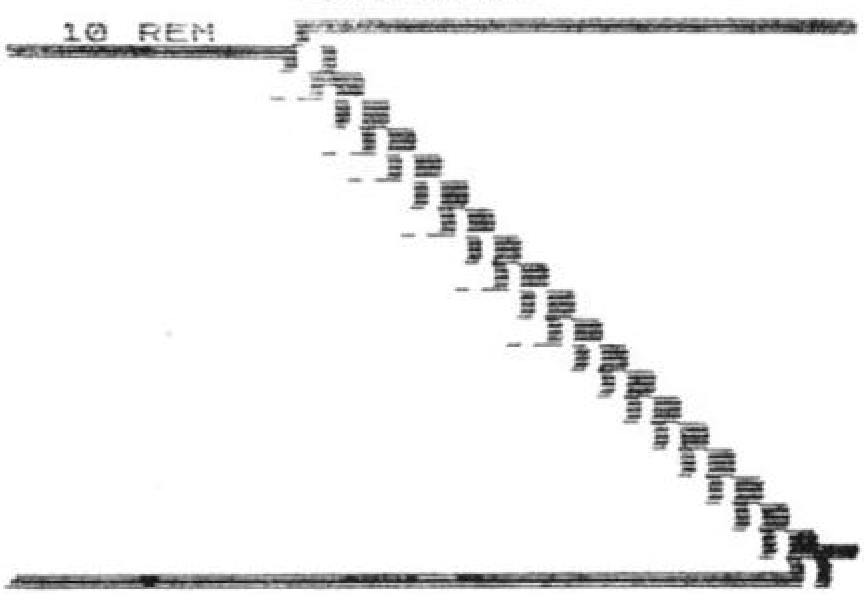
Note that when typing in line 10, the massive REM statement, you have to type in a number of spaces. To check that you have the correct number, enter PRINT PEEK 16511 + 256*PEEK 16512. This should give 686; it can be more, but the extra spaces will not be used. If you POKE, as a direct command, 16514 and 16515 with 118, you won't be able to see the listing.

Lines in inverse are:

999	REM LOAD FROM FIRST 'REM'
1999	REM LOAD INTO FIRST REM
2000	PRINT AT 0,0;"CONFIRM: SAVE SCREEN (Y/N)? . . .
2150	PRINT AT 0,0;"PLEASE CHECK LEADS AND VOLUME ON YOUR TAPE RECORDER"
2999	REM CLEAR SCREEN
3000	PRINT AT 0,0;"CONFIRM: CLEAR SCREEN (Y/N)? . . .
3999	REM STOP PROGRAM
4000	PRINT AT 0,0;"CONFIRM: STOP PROGRAM (Y/N)? . . .
4500	PRINT AT 0,0;"CONFIRM: . . .
4999	REM INITIALIZE INSTRUCTIONS

```
SKETCHPAD

  10 REM
```



```
  20 GOTO 5000
  30 CLS
  40 PRINT B$
  50 PRINT "
```



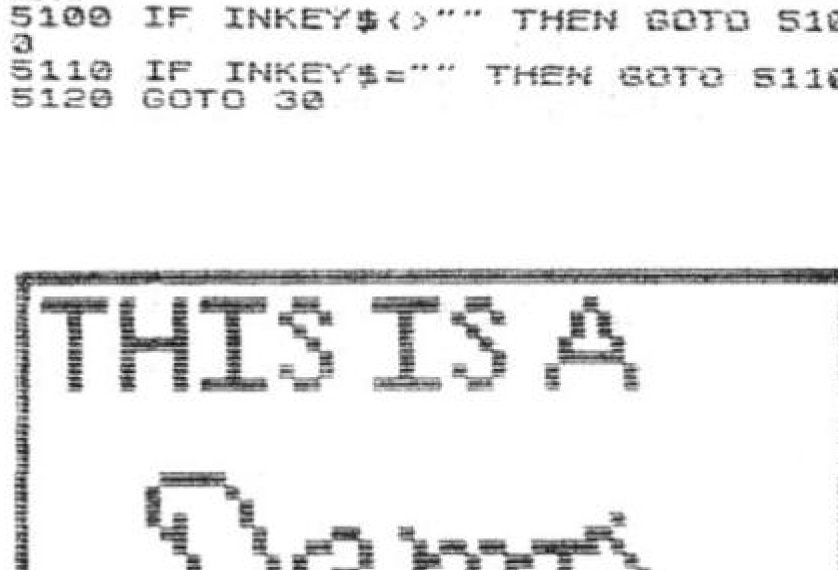
```
  60 FOR I=39 TO 2 STEP -1
  70 PLOT 0,I
  80 PLOT 63,I
  90 NEXT I
 100 PRINT "
```



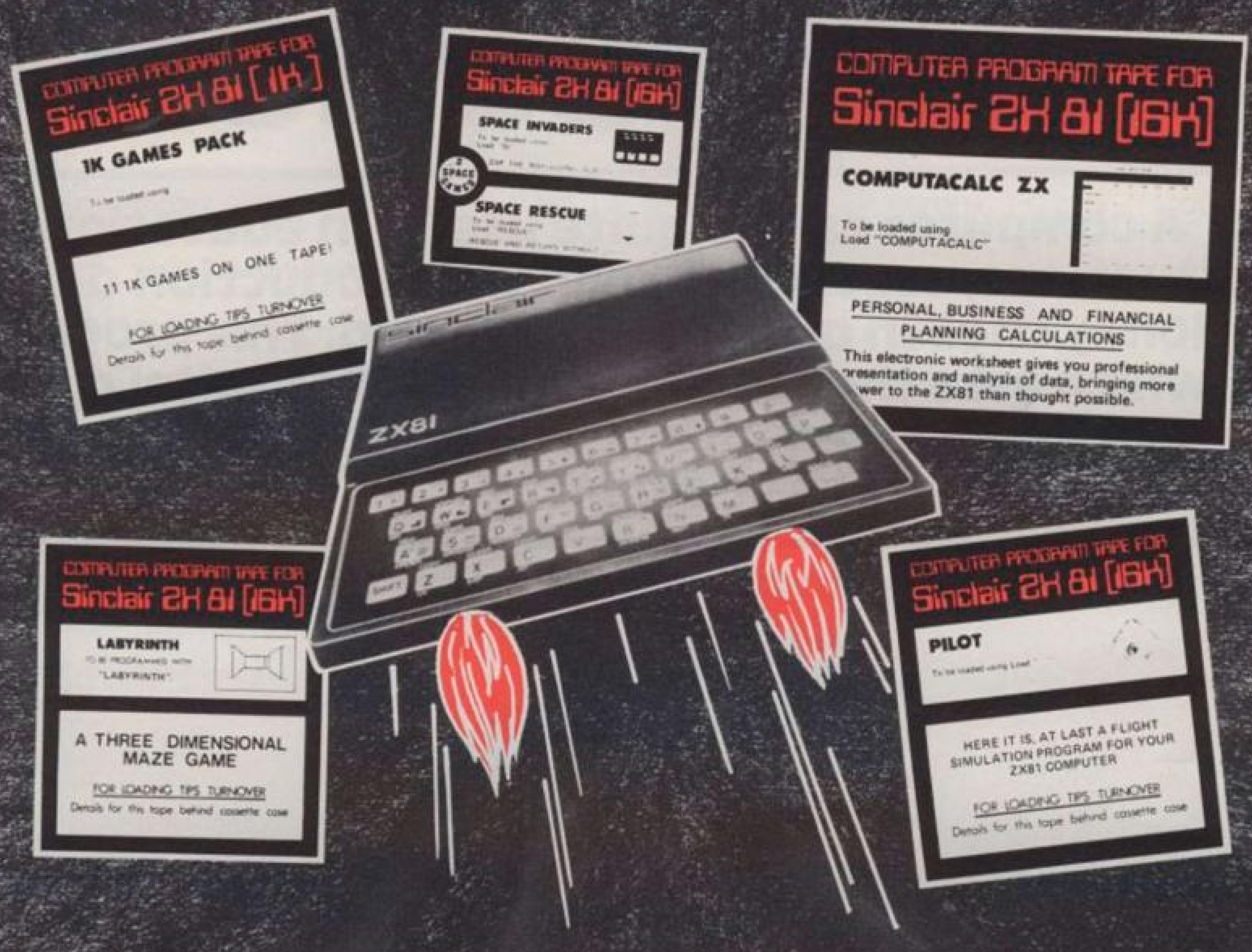
```
 110 LET X=30
 120 LET Y=20
 130 LET M=9
 140 IF M=0 THEN UNPLOT X,Y
 150 IF M=9 THEN PLOT X,Y
 160 LET A$=INKEY$
 170 LET M=M+(9 AND A$="9" AND M
=0)-(9 AND A$="0" AND M=9)
```

```
 180 LET X=X+((A$="E" OR A$="D"
OR A$="C") AND X<62)-((A$="Q" OR
 A$="A" OR A$="Z") AND X>1)
 190 LET Y=Y+((A$="E" OR A$="Q"
OR A$="W") AND Y<30)-((A$="X" OR
 A$="C" OR A$="Z") AND Y>1)
 200 IF A$="1" THEN GOTO 2000
 210 IF A$="2" THEN GOTO 3000
 220 IF A$="3" THEN GOTO 4000
 225 IF A$="4" THEN GOTO 4500
 230 IF M=0 THEN PLOT X,Y
 240 IF M=9 THEN UNPLOT X,Y
 250 GOTO 140
 999 REM [illegible]
1000 CLS
1010 FAST
1020 FOR I=0 TO 19
1030 FOR N=0 TO 31
1040 POKE (Z+I*33+N),PEEK (16516
+I*33+N)
1050 NEXT N
1060 NEXT I
1070 SLOW
1080 PRINT AT 0,0;B$
1090 GOTO 110
1999 REM [illegible]
2000 PRINT AT 0,0;"[illegible]:SAVE
SCREEN (Y/N)?
2010 GOTO 2010+(10 AND INKEY$="N
")+(30 AND INKEY$="Y")
2020 PRINT AT 0,0;B$
2030 GOTO 160
2040 FAST
2050 FOR I=0 TO 19
2060 FOR N=0 TO 31
2070 POKE (16516+I*33+N),PEEK (Z
+I*33+N)
2080 NEXT N
2090 NEXT I
2100 SLOW
2110 PRINT AT 0,0;"SCREEN SAVED,
PLEASE PRESS A KEY
                    "
2120 IF INKEY$="" THEN GOTO 2120
2130 PRINT AT 0,0;"DO YOU WANT I
T ON TAPE (Y/N)?
                    "
2140 GOTO 2140+(60 AND INKEY$="N
")+(10 AND INKEY$="Y")
2150 PRINT AT 0,0;"[illegible]
[illegible]
[illegible]        "
2160 PAUSE 200
2170 PRINT AT 0,0;"TYPE IN NAME
OF PROGRAM,PRESS    RECORD AND PL
AY,THEN NEWLINE
                    "
2180 INPUT N$
2190 SAVE N$
2200 PRINT AT 0,0;B$
2210 GOTO 230
2999 REM [illegible]
3000 PRINT AT 0,0;"[illegible]:CLEAR
 SCREEN (Y/N)?
                    "
3010 GOTO 3010+(10 AND INKEY$="N
")+(30 AND INKEY$="Y")
3020 PRINT AT 0,0;B$
3030 GOTO 230
3040 CLS
3050 GOTO 30
3999 REM [illegible]
4000 PRINT AT 0,0;"[illegible]:STOP
PROGRAM (Y/N)?
4010 GOTO 4010+(10 AND INKEY$="N
")+(30 AND INKEY$="Y")
4020 PRINT AT 0,0;B$
4030 GOTO 230
4040 CLS
4050 PRINT AT 9,10;"SKETCH PAD",
,TAB 4;"TO RUN TYPE RUN,NEWLINE"
4060 STOP
4500 PRINT AT 0,0;"[illegible]:TRANS
FER SCREEN TO THE  PRINTER (Y/N)
?                   "
4510 GOTO 4510+(10 AND INKEY$="Y
")+(50 AND INKEY$="N")
4520 PRINT AT 0,0;"WHAT IS THE T
ITLE ?(TYPE IN,AND PRESS NEWLINE
)                   "
4530 INPUT C$
4540 IF LEN C$<30 THEN GOTO 4590
4550 PRINT AT 0,0;"SORRY,TOO LON
G.PLEASE RE-TYPE A SHORTER TITLE
 "
4560 FOR I=1 TO 50
4570 NEXT I
4580 GOTO 4520
4590 LPRINT TAB (16-(LEN C$/2));
C$
4600 LPRINT
4610 PRINT AT 0,0;"

  "
4620 COPY
4625 LPRINT
4630 PRINT AT 0,0;B$
4640 GOTO 160
4999 REM [illegible]
5000 SLOW
5005 LET Z=PEEK 16396+256*PEEK 1
6397+67
5010 LET B$="1:-SAVE SCREEN  2:-
CLEAR SCREEN 3:-STOP PROGRAM 4:-
PRINTER         "
5020 PRINT TAB 8;"SKETCH PAD"
5030 PRINT ,,"USING THIS PROGRAM
,YOU CAN DRAW ON THE SCREEN,CLEA
R THE SCREEN, SAVE THE PICTURE O
N TAPE,AND      TRANSFER THE SCREE
N TO PRINTER"
5040 PRINT ,,"THE KEYS TO MOVE T
HE FLASHING     CURSOR ARE:"
5050 PRINT TAB 7;"Q  W  E";TAB 8
;"[illegible]";TAB 7;"A<-*->D";TAB 8;"
[illegible]";TAB 7;"Z  X  C"
5060 PRINT ,,"YOU CAN ALSO CHANG
E THE MODE OF DRAWING (I.E.A LIN
E OR NO LINE) BY PRESSING 9 AND
0"
5070 PRINT ,,"ANY PROGRAM TO LOA
D (Y/N)?"
5080 GOTO 5080-(4030 AND INKEY$=
"Y")+(10 AND INKEY$="N")
5090 PRINT ,,"PRESS ANY KEY TO D
RAW"
5100 IF INKEY$<>"" THEN GOTO 510
0
5110 IF INKEY$="" THEN GOTO 5110
5120 GOTO 30
```



BLAST-OFF YOUR ZX81



And explore a galaxy of NEW software!

All these titles are available now from your local Bookstore or Computershop. In case of difficulty, send cost of tape plus 25p postage per tape with the coupon to: Software Masters Ltd., 30 Lincoln Road, Olton, Birmingham B27 6PA. Trade enquiries welcome!

SM2015......	Breakout	16K	5.95
SM2011......	Computacalc	16K	7.95
SM2022......	Constellation	16K	8.00
SM2023......	Football Manager	16K	7.95
SM2003......	Labyrinth	16K	5.95
SM2025......	Magnus	16K	9.95
SM2021......	Mazogs	16K	10.00
SM2017......	Mugsy	16K	5.95
SM2018......	Murgatroyds	16K	5.95
SM2019......	Murgatroyds Revenge	16K	5.95
SM2005......	Nightmare Park/Music	16K	6.95
SM2027......	P.E.P.	16K	5.95
SM2012......	Personal Banking System	16K	9.95
SM2024......	Pilot	16K	5.95
SM2020......	Progmerge	16K	5.95
SM2026......	Puckman	16K	5.95
SM2008......	Space Intruders	16K	5.95
SM2014......	Space Invaders/Space Rescue	16K	6.95
SM2013......	Star Trek/3D Os & Xs	16K	6.95
SM2007......	ZX 81 Chess	16K	6.90
SM2001......	Bumper 7	1K	5.95
SM2016......	1K Games Pack	1K	6.00
SM2002......	1K Super Trio	1K	7.95

Prices include VAT

Please send me the following programs

Code	Title	Cost
		£

Postage:-

I enclose Postal Order/Cheque for Total £

Name

Address

ZXC

Software Masters Ltd.

30 Lincoln Road, Olton, Birmingham B27 6PA, England. Telephone: 021-707 7544.

Computer aided instruction

A computer can be quite useful in educational situations, especially when the production of randomly chosen questions is needed. This program which asks the student to identify capital cities can easily be adapted to deal with a variety of subjects.



This program picks 10 countries at random, asking the user in each case to name the capital city of that country. It gives a score out of 10 at the end of the round (line 430) then gives the user the option of either terminating the run, or of going through another 10 questions.

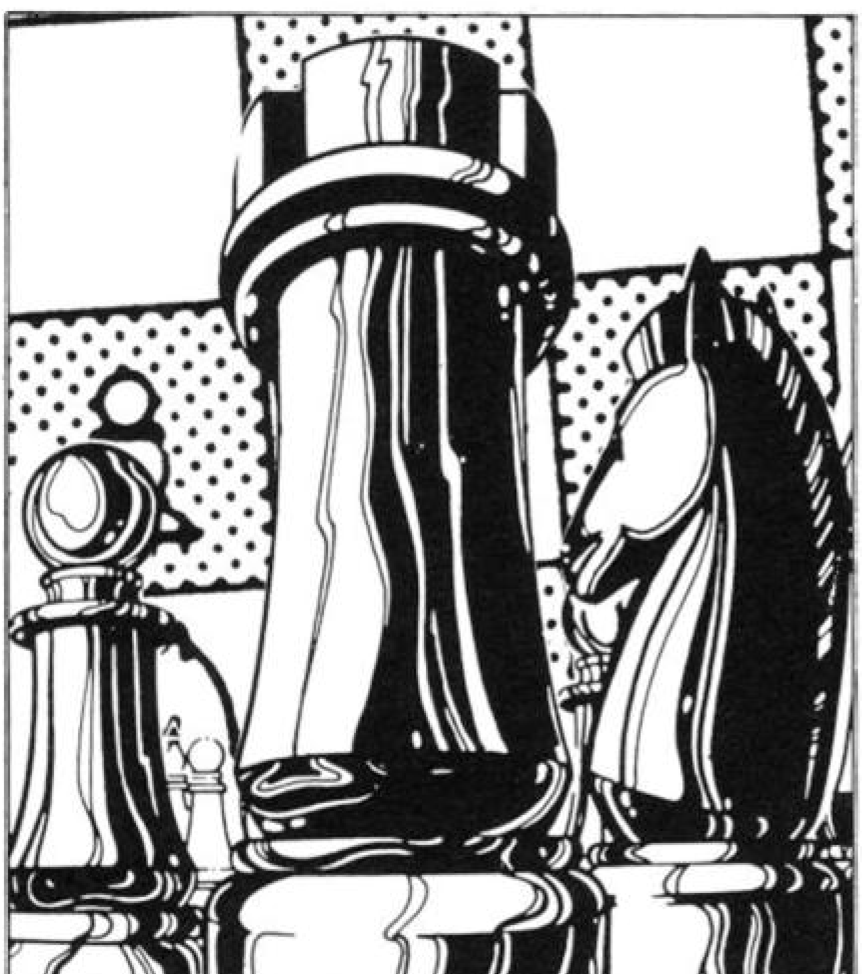
There is no mechanism within the program to ensure that the same city is not asked for more than once in a run. The program does, however, give the correct answer if the student was wrong.

To adapt the program for other subjects, change the specific question asked (the routine from line 180) and — of course — the 'questions' and 'answers' given from line 1040.

```
  10 REM CAPITALS OF THE WORLD
  20 REM (C) HARTNELL 1982
  25 LET SCORE=0
  30 SCROLL
  40 PRINT "I WILL NAME 10 COUNT
RIES AND"
  50 SCROLL
  60 PRINT "YOU HAVE TO NAME THE
IR CAPITALS"
  70 SCROLL
  80 PRINT "AT THE END YOU WILL
BE GIVEN A"
  90 SCROLL
 100 PRINT "SCORE OUT OF 10."
 110 SCROLL
 120 SCROLL
 130 PRINT "PRESS NEWLINE/RETURN
WHEN"
 135 SCROLL
 140 PRINT "YOU ARE READY TO GO.
.."
 150 INPUT U$
 155 SCROLL
 156 SCROLL
 160 FOR A=1 TO 10
 165 SCROLL
 167 PRINT "QUESTION NUMBER ";A;
"; -"
 170 GOSUB 1000
 175 SCROLL
 180 PRINT "WHAT IS THE CAPITAL"
 190 SCROLL
 200 PRINT TAB 6;"OF ";A$;"?"
 210 INPUT U$
 215 SCROLL
 220 IF U$=B$ THEN PRINT "YES, "
;B$;" IS CORRECT"
 230 IF U$=B$ THEN LET SCORE=SCO
RE+1
 240 IF U$<>B$ THEN PRINT "NO, T
HE CAPITAL OF"
 250 SCROLL
 260 IF U$<>B$ THEN PRINT A$;" I
S ";B$
 262 SCROLL
 265 IF U$<>B$ THEN PRINT TAB 6;
"NOT ";U$;"."
 270 SCROLL
 280 SCROLL
 290 IF A<10 THEN PRINT "YOUR SC
ORE SO FAR IS ";SCORE;" OUT OF "
;A
 300 SCROLL
 310 SCROLL
 320 PRINT "STAND BY"
 330 FOR G=1 TO 24
 340 SCROLL
 350 FOR H=1 TO 5
 355 NEXT H
 360 NEXT G
 370 NEXT A
 380 SCROLL
 390 SCROLL
 400 SCROLL
 410 PRINT "YOUR TOTAL SCORE WAS
"
 420 SCROLL
 430 PRINT TAB 4;SCORE;" OUT OF
10."
 440 SCROLL
 450 SCROLL
 460 PRINT "DO YOU WANT ANOTHER
GO?"
 470 INPUT U$
 475 CLS
 480 IF CODE (U$)<>CODE "N" THEN
 RUN
 490 SCROLL
 500 PRINT "OK, BYE FOR NOW"
 510 STOP
1000 REM *****************
1010 LET K=10*(INT (RND*20)+4)+1
000
1020 GOSUB K
1030 RETURN
1040 LET A$="CAMBODIA"
1042 LET B$="PHNOM PENH"
1045 RETURN
1050 LET A$="BOLIVIA"
1052 LET B$="SUCRE"
1055 RETURN
1060 LET A$="CUBA"
1062 LET B$="HAVANA"
1065 RETURN
1070 LET A$="AFGHANISTAN"
1072 LET B$="KABUL"
1074 RETURN
1080 LET A$="ANGOLA"
1082 LET B$="LUANDA"
1085 RETURN
1090 LET A$="AUSTRIA"
1092 LET B$="VIENNA"
1095 RETURN
1100 LET A$="VIETNAM"
1102 LET B$="HANOI"
1105 RETURN
1110 LET A$="URUGUAY"
1112 LET B$="MONTEVIDEO"
1115 RETURN
1120 LET A$="UGANDA"
1122 LET B$="KAMPALA"
1125 RETURN
1130 LET A$="THAILAND"
1132 LET B$="BANGKOK"
1135 RETURN
1140 LET A$="SWITZERLAND"
1142 LET B$="BERN"
1145 RETURN
1150 LET A$="SRI LANKA"
1152 LET B$="COLOMBO"
1155 RETURN
1160 LET A$="SPAIN"
1162 LET B$="MADRID"
1165 RETURN
1170 LET A$="ROMANIA"
1172 LET B$="BUCHAREST"
1175 RETURN
1180 LET A$="PANAMA"
1182 LET B$="PANAMA"
1185 RETURN
1190 LET A$="MEXICO"
1192 LET B$="MEXICO CITY"
1195 RETURN
1200 LET A$="VENEZUELA"
1202 LET B$="CARACAS"
1205 RETURN
1210 LET A$="JAMAICA"
1212 LET B$="KINGSTON"
1215 RETURN
1220 LET A$="DENMARK"
1222 LET B$="COPENHAGEN"
1225 RETURN
1230 LET A$="ECUADOR"
1232 LET B$="QUITO"
1235 RETURN
1240 LET A$="FIJI"
1242 LET B$="SUVA"
1245 RETURN
```

The chess giants grapple

Sinclair Research is selling a chess program for the ZX81, developed by Psion/Microgen. Reviewer Nick Pearce tried it out, and played it off against Artic's Chess 11.



Originally I intended just to review the Psion program, as part of my review of the new Sinclair ZX81 software, but decided that a comparison with ZX Chess 11 by Artic, fast becoming the standard for ZX chess, would be more appropriate. The comparison, although interesting, is perhaps a little unfair. Chess 11 currently retails at £9.95, while the Psion program, together with CHESS CLOCK on the other side of the cassette, sells for £3.00 less.

Both programs display the board on the screen, and pieces are represented by letters, K for King, Q for Queen, B for Bishop and so on, with the colour of the piece being the colour of the displayed letter, inverse letters for black pieces.

Both games use the standard algebraic system of chess notation for the board co-ordinates, although the Artic program is the only one which actually puts these co-ordinates on the screen.

The Artic board is always displayed the same way up, no matter which end the human is playing from, while the Psion board is shown with the player at the bottom. I feel this makes the Psion program rather easier to play.

Although I expected to, I did not find the lack of co-ordinates on the screen a disadvantage, as I found with both programs it was only really possible to play a normal game by having a proper board set up beside the TV anyway.

Both programs give you the option of setting up a game position. The Artic program requires you to set up a piece by first defining the co-ordinates, then entering a code for the piece you want there. On the Psion board, you use a cursor (=) to move pieces around the board.

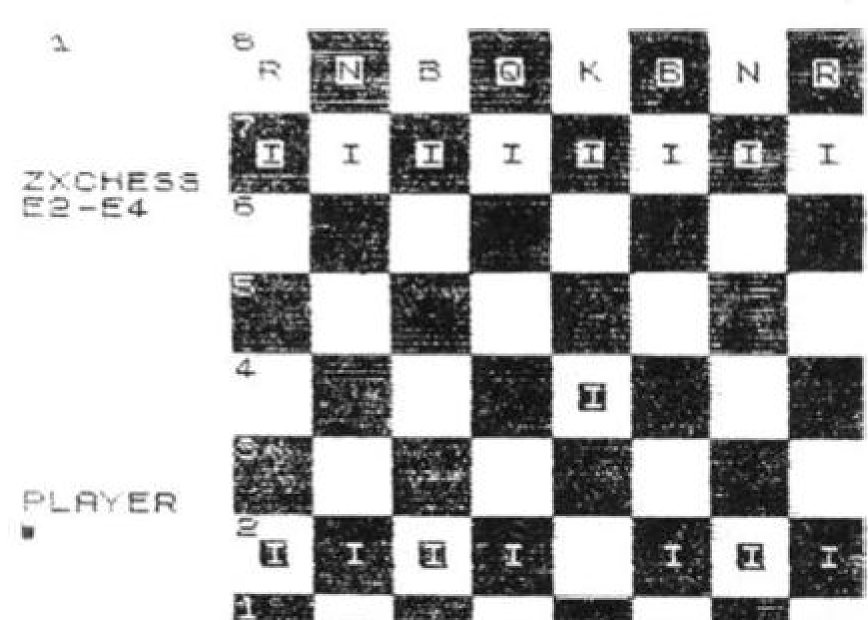
A deficiency with the Psion program for the serious player is the lack of a 'game save' facility, which the Artic program includes. Psion's also lacks the ability to print out a copy of the screen at any time, or a list of moves.

Against this, I particularly liked the Psion feature of being able to resign at any time to start a new game. If you get fed up with a game on the Artic program, you have to either continue to the bitter end, or switch off and reload to start a new game.

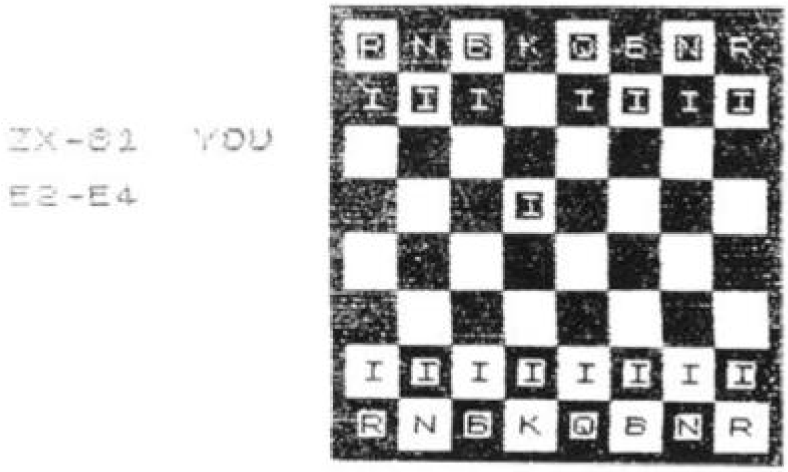
So much for the immediately apparent differences between the two programs. I'll look now at how well they actually play.

Both allow castling and en-passant moves, and neither permits illegal moves. I do not play a particualrly good chess game, and consequently found both programs hard to beat, even on the lowest levels.

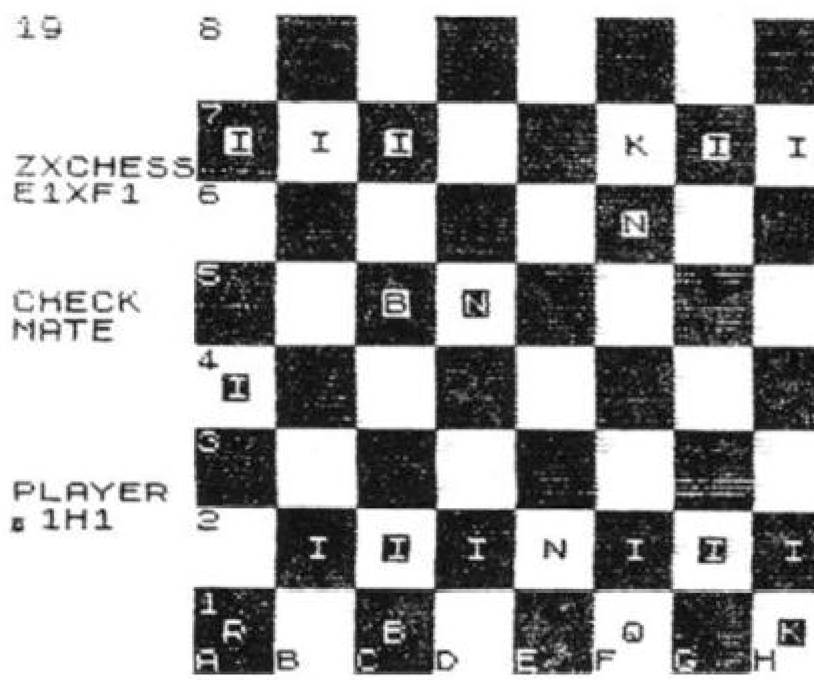
Both games seemed to go for check whenever possible, even if they were not in a position to sustain an effective attack. During one game with the Psion program,



the program repeatedly went for check from a losing position, and in so doing sacrificed its remaining valuable pieces. This left me with an overwhelming piece advantage, from which I was able to win easily. However, this was a rather unusual finish to a game against the Psion program. In most games I played, the Psion program put up a good, solid defence, and was not easy to beat.



01	E2-E4	E7-E5	26	C3-B4	E8-C8
02	G1-F3	B8-C6	27	D4-D5	B8-A8
03	F1-B5	F8-D6	28	E2-D2	F7-G6
04	B5XC6	D7XC6	29	E3-G3	G6-F7
05	B1-C3	C8-G4	30	G3-G4	C8-D8
06	H2-H3	G4XF3	31	G4-H5	F7-G8
07	D1XF3	G8-F6	32	H5-F3	A6-B6
08	E1-G1	E8-G8	33	B4-E7	D8-E8
09	D2-D3	D6-E7	34	E7-A3	B5-B4
10	C1-D2	D8-D6	35	A3-B2	D7-C5
11	A1-E1	F8-D8	36	D3-E3	A8-B8
12	E1-E2	D6-B4	37	G1-F1	B6-A6
13	C3-D5	B4-D6	38	F1-E1	G7-G5
14	D5XE7	D6XE7	39	E1-D1	G8-G7
15	A2-A3	E7-E6	40	D1-E1	E8-F8
16	F1-E1	B7-B6	41	E1-D1	G7-G6
17	D2-C3	A8-B8	42	D1-E1	F8-D8
18	F3-G3	D8-E8	43	F3-F5	G6-G7
19	G3-E3	F6-D7	44	B2XE5	F6XE5
20	D3-D4	F7-F6	45	F5XG5	G7-F7
21	E1-D1	G8-F7	46	E3-F3	F7-E8
22	D1-D3	E6-C4	47	G5XE5	E8-D7
23	B2-B3	C4-A6	48	F3-F7	D7-C8
24	A3-A4	B6-B5	49	F7XC7	
25	A4XB5	C6XB5			



In order to obtain a comparison between the two programs, a friend lent me his ZX81, and I played one program against the other.

The levels of play of each program are related to the time taken to respond to a move, and are not directly compatible between the two programs.

I rang the changes between the three easiest Artic, and the two easiest Psion, levels of play.

I must stress that this comparison is a little unfair, as the Psion does not claim to play chess to the standard of the more expensive Artic game.

As you may have anticipated, the Artic program played the stronger game. To Psion's credit, three of the ten games were held to a draw. Artic won six, Psion won one.

In a game between Psion on the ZX81, playing white, and Sargon 11 on Tandy, Sargon won, but took some 45 moves to do so. Both played at level one.

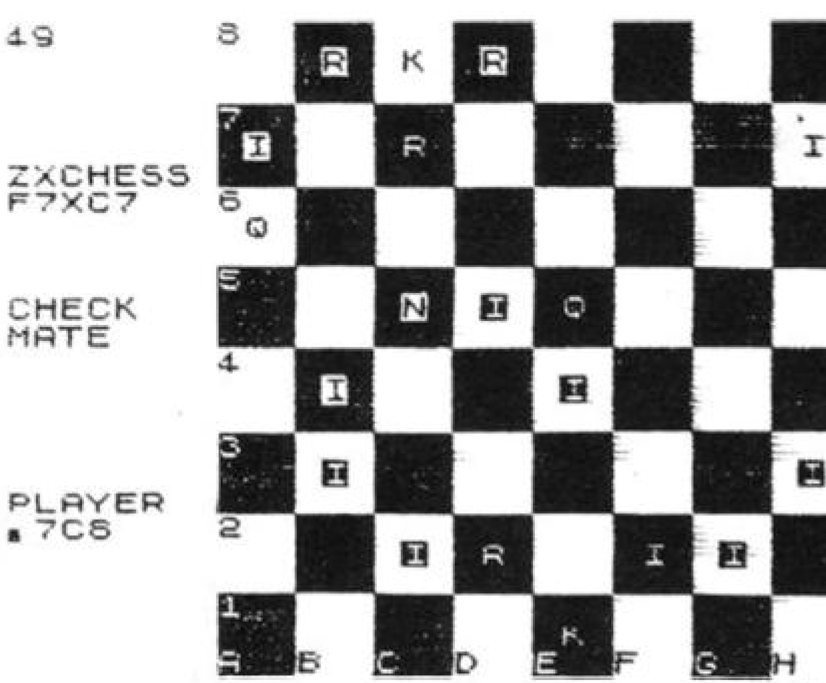
Both the Psion and the Artic chess programs play a reasonable game. The Artic program plays a stronger, more sophisticated game, and with its save and move listing features will be the more attractive for serious users. For the casual player, who wants just an occasional game of chess, Psion is a reasonable alternative. The Psion program is more user-friendly, and I particualrly liked its 'resign' facility.

There is little doubt that most ZX81 owners will learn a bit more about their computers, and the game of chess, through playing against any of these programs.

On the B side of Psion's chess is CHESS CLOCK. This simulates the clock used in tournaments to limit the amount of time taken for each move. Two digital readouts display the time taken by each player.

I thought that CHESS CLOCK was a bit of a gimmick, and unlikely to be used seriously. There might be a few dedicated players who would be prepared to use a ZX81 and a television to timekeep during serious games.

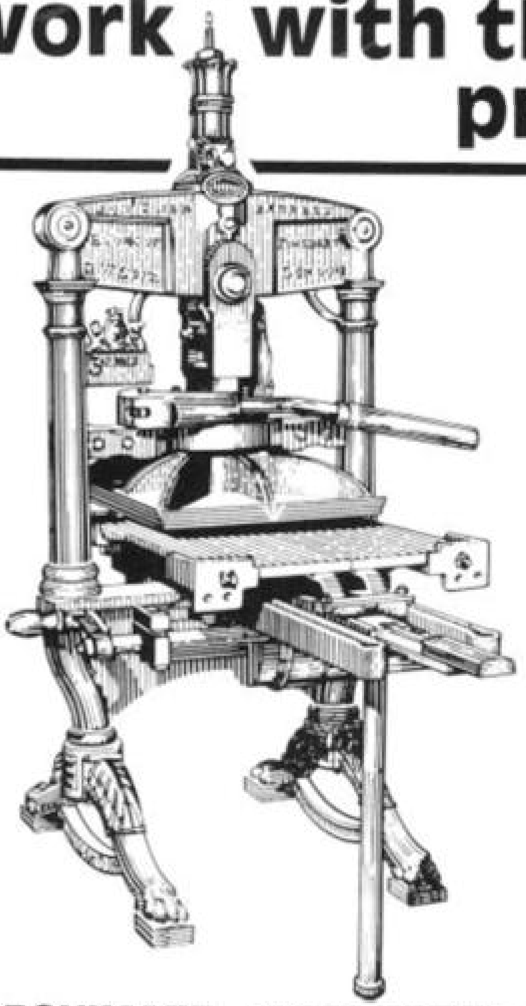
I thought the Sinclair advertisement for CHESS CLOCK somewhat misleading. It cannot be used at the same time as the chess program itself as it implied by the statement ". . . can be used at any time", unless you happen to have two television sets and two ZX81s.



		GAME	LEVEL OF PLAY (TIME <SECS>) PSION	ARTIC	RESULT AND MOVES
P	W	1	1 (05)	0 (02)	PSI-28
S	H	2	1 (05)	1 (15)	ART-34
I	I	3	2 (35)	0 (02)	ART-18
O	T	4	2 (35)	1 (15)	DRAW
N	E	5	2 (35)	2 (40)	DRAW
A	W	6	1 (05)	0 (02)	ART-58
R	H	7	1 (05)	1 (15)	ART-56
T	I	8	2 (35)	0 (02)	DRAW
I	T	9	2 (35)	1 (15)	ART-47
C	E	10	2 (35)	2 (40)	ART-49

Making your ZX work

When you're ZX81 gets tired of zapping Invaders, you can put it to work with the following utility programs.



APPROXIMATE DEFINITE INTEGRAL

This program enables you to work out a definite integral by using Simpson's rule. You follow the prompts given, entering the function to be used in line 30.

```
  5 REM APPROX. DEFINITE
  7 REM           INTEGRAL
 10 REM BY SIMPSON/S METHOD
 20 PRINT "ENTER FUNCTION"
 30 INPUT X$
 40 PRINT X$,"ENTER X0"
 50 INPUT M
 60 PRINT M,"ENTER X2P"
 70 INPUT N
 80 PRINT N;"     NO. OF DIVISION
S?"
 90 INPUT Z
100 LET R=(N-M)/2/Z
110 LET Q=0
120 LET X=M
130 LET Y=VAL (X$)
140 LET Q=Y+Q
150 LET X=X+R
160 LET Y=VAL (X$)
170 LET Q=Y*4+Q
180 LET X=X+R
190 LET Y=VAL (X$)
200 LET Q=Y+Q
210 LET Z=Z-1
212 SCROLL
215 IF Z<>0 THEN PRINT Q*R/3
220 IF Z<>0 THEN GOTO 140
230 LET P=Q*R/3
235 SCROLL
240 PRINT "THE ANSWER IS ";P
```

Sample run:

```
ENTER FUNCTION
X*X*X-3                    ENTER X0
0                          ENTER X2P
1    NO. OF DIVISIONS?

 -0.299975
 -0.5996
 -0.897975
 -1.1936
 -1.484375
 -1.7676
 -2.039975
 -2.2976
 -2.535975

THE ANSWER IS -2.75
```

POISSON DISTRIBUTION

Using this routine, you can determine the probability at a point which we'll call X, and the cumulative probability from zero to X. The sample run given uses a value of 6 for M, and 8 for X. The probability, f, is given at the end, as is the cumulative probability, L(P).

```
 20 PRINT "ENTER M"
 30 INPUT A
 40 PRINT "M= ";A,"ENTER X"
 50 INPUT B
 55 PRINT
 60 PRINT "M= ";A;"    X= ";B
 70 LET Q=EXP (-A)
 80 LET R=Q
 90 IF B=0 THEN GOTO 140
 95 SCROLL
 96 PRINT "P","L(P)"
100 FOR Z=1 TO B
110 LET R=A*R/Z
115 SCROLL
116 PRINT R,
120 LET Q=Q+R
126 PRINT Q
130 NEXT Z
140 SCROLL
142 SCROLL
145 PRINT "P= ";R
147 SCROLL
150 PRINT "L(P)= ";Q
```

Sample run:

```
P                      L(P)
.014872513             .017351255
.044617539             .061968805
.089235078             0.15120388
0.13385262             0.2850565
0.16062314             0.44567964
0.16062314             0.60630278
0.13767693             0.74397976
0.10325773             0.84723749

P= 0.10325773
L(P)= 0.84723749
```

AREA CALCULATOR

This program works out the floor area of a house, in order to determine how much carpet is required to cover it. Just follow the prompts given.

```
  10 REM AREA CALCULATOR
  15 LET TOTAL=0
  17 SCROLL
  20 PRINT "HOW MANY ROOMS?"
  30 INPUT R
  40 FOR Q=1 TO R
  50 SCROLL
  60 PRINT "WHAT IS LENGTH OF RO
OM ";Q;"?"
  70 INPUT L
  75 SCROLL
  80 PRINT "WHAT IS WIDTH OF ROO
M ";Q;"?"
  90 INPUT W
 100 LET A=W*L
 110 SCROLL
 120 PRINT "AREA OF ROOM ";Q;" I
S ";A
 130 LET TOTAL=TOTAL+A
 140 SCROLL
 150 PRINT "AND THE TOTAL AREA "
;
 160 IF Q<R THEN PRINT "SO FAR "
 180 SCROLL
 190 PRINT ,"IS ";TOTAL
 200 SCROLL
 210 SCROLL
 220 NEXT Q
```

Sample run:

```
HOW MANY ROOMS?
WHAT IS LENGTH OF ROOM 1?
WHAT IS WIDTH OF ROOM 1?
AREA OF ROOM 1 IS 180
AND THE TOTAL AREA SO FAR
                IS 180

WHAT IS LENGTH OF ROOM 2?
WHAT IS WIDTH OF ROOM 2?
AREA OF ROOM 2 IS 120
AND THE TOTAL AREA SO FAR
                IS 300

WHAT IS LENGTH OF ROOM 3?
WHAT IS WIDTH OF ROOM 3?
AREA OF ROOM 3 IS 104
AND THE TOTAL AREA
                IS 404
```

TYPING TUTOR

This program generates a letter of the alphabet at random and then gives you a limited time to find it. The program will tell you if you're right, wrong, or have just taken too long. Once you've mastered it in its present form, decrease the length of the a loop (line 100). The listing here is for the Spectrum,. but it is easy to modify for the ZX81. Use upper case A in the loop counter and A$ (you haven't any choice, anyway, on the ZX81); change the colon in lines 130 and 140 into a semi-colon, and replace the word STOP with the letter Q.

```
   5 REM ENGAGE CAPS LOCK FIRST
  10 LET B$=CHR$ (INT (RND*26)+C
ODE "A")
  20 PRINT AT 10,0;"QUICKLY, PRE
SS ";B$
 100 FOR a=0 TO 100
 110 LET a$=INKEY$
 120 IF a$<>"" THEN GO TO 140
 125 NEXT a
 130 PRINT "Time is up": STOP
 140 IF A$=B$ THEN PRINT "You we
re right": STOP
 150 PRINT "You were wrong"
```


ADVERTISEMENT INDEX

A Picturesque is worth a thousand words

A number of 'tool kits' are available to make it possible to write better programs on the ZX81. ZX Computing staff look at two of them, Screen Kit 1 from Picturesque and Graphics toolkit from JRS Software.



Screen kit 1 is an attractively packaged screen toolkit from 'Picturesque' sold by mail order for £5.70. It consists of a number of machine code routines to help a ZX81 user smarten up his graphics in a BASIC program.

The cassette inlay and instruction card are tidily presented in blue, well printed and easy to read. From its smart appearance I assumed it would LOAD first time but I was not so lucky. On playing back the tape through the loudspeaker I discovered the signal was a little noisy and was not recorded at a sufficiently loud volume. After wedging a small piece of paper under the cassette (!) and setting the highest volume level on my cassette recorder it LOADed easily.

Screen Kit 1 has eight facilities four-way scrolls; CLS; Borders; CPS; Reverse; Cursors; Memory and SAVE/LOAD.

The SAVE and LOAD facilities are perhaps the most useful, being at double speed. All the routines are accessed from BASIC via USR statements & POKE commands. The instruction card is A4 size, double sided and folded in half with compressed text printed in Black. In this way Picturesque have managed to get some pretty comprehensive and detailed instructions onto a single sheet.

The first four commands scroll the screen in the appropriate direction. One POKE adjusts how many scrolls will be actioned before returning to BASIC. For instance, to scroll the screen five times to the left simply POKE 16661,5 then RAND USR 16660. The unfortunate thing about the scrolls in Screen Kit 1 is that they do not wrap-around ie the characters lost at the edges of the screen do not reappear at the opposite edge. Wrap-around scrolls are far more useful in that you can do continuous backgrounds that are constantly moving but do not need to be updated because it wraps around the screen. The principal is quite similar to a 1920s westerns where the Sheriff would run on a conveyor belt and behind him would be the background painted on a giant canvas sheet that



would be rotated like a belt (thus one always saw the same scenery go past again and again!)

Clearly Crashed

Screen Kit 1 also provides a CLS facility which can be used to overcome the slowness of the ROM's routine. Instead of the usually slow CLS where the characters can be seen to be changed spaces from top to bottom, the Picturesque CLS is good, instant clear screen which is far tidier. CLS also has a second use. If you poke 16863 with a character code, that character will be used instead of spaces in the CLS routine. Unfortunately Screen Kit 1 has no error checking routine at all so if you POKE an invalid character code into 16863 and then access the CLS routine then the program crashes. This sort of defect can be very annoying and doesn't show a very thorough approach to the product.

The BORDERS routine is quite impressive in that it is a fast way of drawing boxes on the screen. You have to specify the line and column position of the top left hand corner and also the height and width inside the border. This requires four POKES which is really quite a lot of typing for a single command. CPS is simply a CLS for a specified part of the screen, the POKEs are the same as those used in the Border command. Clearing part of the screen is not something that is a vital aid to the Graphics programmer. It does have its uses but not *that* many.

The next command is a useful function. It changes part of the screen into inverse video, which in loop would make a very straightforward way of highlighting areas of the screen but for the five lines of BASIC code required every time this operation is needed. The next command is called CURSOR and here is a quote from the manual: "Simulates INPUT, but gives you a flashing cursor". What actually happens is that it flashes a black blob on the screen which disappears when a key is pressed. The character code of the key pressed is returned. To say that it simulates INPUT is not quite accurate; more correctly, it could be described as INKEY$ with flashing blob!

Typing PRINT USR 16886 returns a figure which is the free memory in bytes. The SAVE & LOAD routines are the most useful part of the package. They operate at double speed and are used for storage of variables on tape. No file names are allowed so that rules out a tape filing system for a database program, where it could load up a selected file from, say, twenty on one tape.

This is a pity since the double speed SAVE & LOAD in their machine code monitor 'ZC-MC' allow single letter file names. When I tried the routines for Sinclair's SAVE & LOAD and a similar pattern appeared on the screen, except the bars were narrower. The SAVE & LOAD routines are a reason to buy the package in themselves.

Plotting Power

The SAVE and LOAD routines are very good and very useful, being at double speed for the variables, but bear little relation to the title Screen Kit 1. Perhaps they should have called it 'Programmers toolkit'. As Screen Kit 1 is stored in a REM statement it becomes part of your program, which has the advantage that it doesn't need to be loaded above RAMTOP every time you want to use it. But it does mean that if you start writing a program and decide you need Screen Kit 1 and haven't already LOADed it, then to merge it with your own program is a very complicated process described in 27 steps in the instrucions

Despite this, it is a very useful package overall, and one which will enable you to produce much more professional looking programs.

Graphic Toolkit

Graphics Toolkit is sold by mail order from JRS Software at a cost of £5.95. This includes VAT and Postage. Like Screen Kit 1, it consists of a number of machine code routines to help the BASIC programmer smarten up his graphics.

The cassette is a studio produced type and the quality is of a good standard because the signal is 'processed' by the studio. The packaging is not as smart as that of Picturesque, but the instructions consist of two sheets of double sided, compressed A4 as opposed to one. They go to the same depth as Picturesque's instructions but cover an extra eleven functions.

Graphics Toolkit has twenty three routines:

DRAW	UNDRAW
FOREGROUND ON	FOREGROUND OFF
BORDER	UNBORDER
FILL	REVERSE
UP	DOWN
LEFT	RIGHT
EDITPRINT	UPSCROLL
DOWNSCROLL	LEFTSCROLL
RIGHTSCROLL	ONSCREEN
OFFSCREEN	BACKGROUND ON
BACKGROUND OFF	SEARCH & REPLACE
SQUARE	

The most powerful and useful feature is the DRAW command. It allows the user to define a multicharacter shape in a REM statement and then DRAW or UNDRAW it at any point on the screen. It will draw a shape of any size instantly and so is ideal where fast moving graphics are required. The shape is defined in a REM using characters, (the ones to be printed), and direction codes (to indicate where each character is to be placed). For instance, to define a simple square:

10 REM A SQUARE: ■

The user may place his own label first (in this case it is 'A SQUARE') then there is a colon to tell Graphics Toolkit that the definition is starting. Then comes the first character, followed by a < (greater than) indicating that the next character is to be placed to the right of the previous one. A less than sign means move left, a 'V' means move down and an 'A' means move up. The FOREGROUND ON/OFF facility provides an added effect when used with DRAW & UNDRAW. When foreground mode is on, the shapes that are moved around the screen appear to move behind graphics already on the screen.

A demonstration is provided on the cassette, in which a predefined snake moves behind a cactus. Any number of shapes can be defined and used because one POKE before the DRAW command sets the line number of the appropriate shape. The BORDER command is not the same as the Screen Kit. Instead of drawing a one pixel wide border anywhere on the screen, it draws an instant border around the edge of the screen using any character.

Only the bottom line position is variable. This is so you have the choice of using all 24 lines of the screen or leaving the bottom two free as a sort of 'text window'. UNBORDER simply removes the border. EDITPRINT allows you to use that window; it moves the print position to the top edit line which is normally inaccessible from PRINT.

Giving Ground

FILL does exactly what it says. It fills a portion of the screen with a chosen character. The user, by way of two POKEs decides the line to start filling from and how many lines to fill. On this, as on all commands there is full error checking so that if you try to use an invalid character or FILL off the bottom of the screen it gives an error code in the usual format Code/Line No. An actual error message system would have been more helpful but codes are better than nothing. REVERSE is similar to Screen Kit 1's command of the same name, in that it changes part of the screen to inverse Video.

FOREGROUND mode affects both FILL & REVERSE; if Foreground mode is on, only Foreground characters will be changed. Foreground characters count as anything which isn't a Background character. Sounds confusing, doesn't it?

Background is set by BACKGROUND ON. This clears the screen to character of your choice and selects it as your Background. That is similar to the PAPER system on the Spectrum. The Foreground characters can be anything else. For instance, you have 30 inverse asterisks randomly placed on the screen, the Background being fullstops.

When a REVERSE command is done with Foreground on, the asterisks will be changed to ordinary asterisks but the fullstops would be left unchanged.

The Scrolls wrap-around the screen, making some interesting effects possible. I created the effect of moving through space by randomly printing fullstops on the screen, REVERSEing the whole screen and then just repeatedly DOWNSCROLLing it, ideal for a space game!

An interesting facility is ONSCREEN & OFFSCREEN. They turn the screen on and off respectively, not as in FAST mode but by a clever technique which uses no extra memory.

As they are instant, the screen can be flashed in a loop to give a quite stunning effect. SEARCH & REPLACE searches on the screen for a chosen character and replaces it with another character. This would take quite sometime in BASIC but with Graphics Tookit it operates immediately. The uses of this are not obvious, nor varied, but really depend on the user's imagination. SQUARE is quite similar to Screen Kit 1's BORDER oommand but far slower.

Conclusion

Graphics Toolkit has far more routines than Screen Kit 1 but lacks double speed LOAD & SAVE. Good value at £5.95.

Don't let its size fool you.

If anything NewBrain is like the Tardis.

It may look small on the outside, but inside there's an awful lot going on.

It's got the kind of features you'd expect from one of the really big business micros, but at a price of under £200 excluding VAT it won't give you any sleepless nights.

However, let the facts speak for themselves.

You get what you don't pay for.

NewBrain comes with 24K ROM and 32K RAM, most competitors expect you to make do with 16K RAM.

What's more you can expand all the way up to 2 Mbytes, a figure that wouldn't look out of place on a machine costing ten times as much.

We've also given you the choice of 256, 320, 512 and 640 x 250 screen resolution, whereas most only offer a maximum of 256 x 192.

Big enough for your business.

Although NewBrain is as easy as ABC to use (and child's-play to learn to use) this doesn't mean it's a toy.

Far from it.

It comes with ENHANCED ANSI BASIC, which should give you plenty to get your teeth into.

And it'll also take CP/M® so it speaks the same language as all the big business micros, and feels perfectly at home with their software.

NO OTHER MICRO HAS THIS MUCH POWER IN THIS MUCH SIZE FOR THIS MUCH MONEY.



So as a business machine it really comes into its own.

The video allows 40 or 80 characters per line with 25 or 30 lines per page, giving a very professional 2000 or 2400 characters display in all on TV and/or monitor. And the keyboard is full-sized so even if you're all fingers and thumbs you'll still be able to get to grips with NewBrain's excellent editing capabilities.

When it comes to business graphics, things couldn't be easier. With software capabilities that can handle graphs, charts and computer drawings you'll soon be up to things that used to be strictly for the big league.

Answers a growing need.

Although NewBrain, with its optional onboard display, is a truly portable micro, that doesn't stop it becoming the basis of a very powerful system.

The Store Expansion Modules come in packages containing 64K, 128K, 256K or 512K of RAM. So, hook up four of the 512K modules to your machine and you've got 2 Mbytes to play with. Another feature that'll come as a surprise are the two onboard V24 interfaces.

With the aid of the multiple V24 module this allows you to run up to 32 machines at once, all on the same peripherals, saving you a fortune on extras.

The range of peripherals on offer include dot matrix and daisy wheel printers, 9," 12" and 24" monitors plus 5¼" floppy disk drives (100 Kbytes and 1 Mbyte) and 5¼" Winchester drive (6-18 Mbytes).

As we said, this isn't a toy.

It doesn't stop here.

Here are a couple of extras that deserve a special mention.

The first, the Battery Module, means you won't be tied to a 13 amp socket. And, even more importantly, it means you don't have to worry about mains fluctuations wreaking havoc with your programs.

The ROM buffer module gives you a freedom of another sort.

Freedom to expand in a big way. It gives you additional ROM slots, for system software upgrades such as the Z80 Assembler and COMAL, 2 additional V24 ports, analogue ports and parallel ports.

From now on the sky's the limit.

Software that's hard to beat.

A lot of features you'd expect to find on software are actually built into NewBrain so you don't need to worry about screen editing, maths, BASIC and graphics.

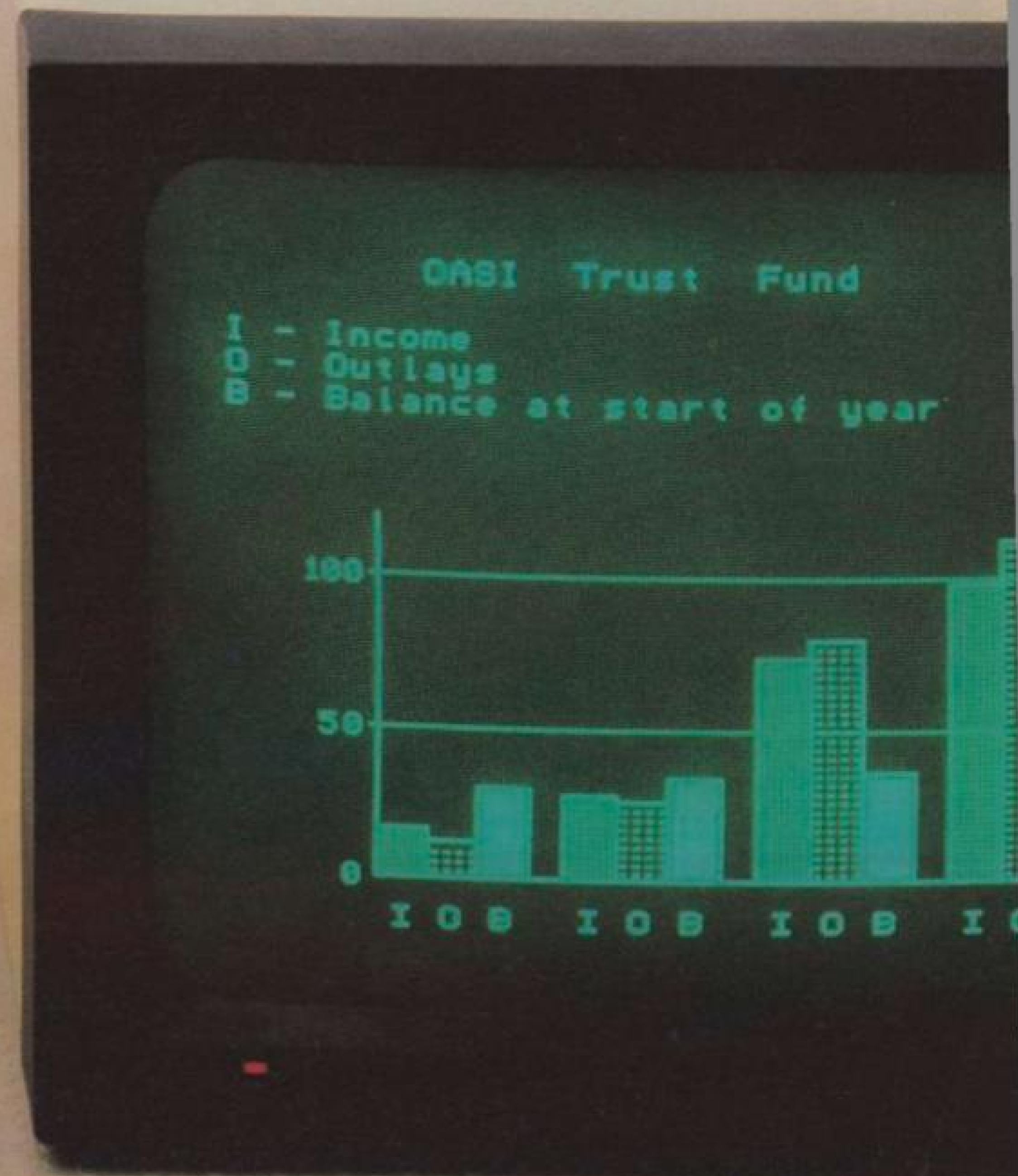
However, if you're feeling practical you can always tackle household management, statistics and educational packages. And because NewBrain isn't all work and no play, there's the usual range of mind-bending games to while away spare time.

Waste no more time.

To get hold of NewBrain you need go no further than the coupon at the bottom of the page.

With your order we'll include a hefty instruction manual so you'll know where to start, and a list of peripherals, expansion modules, and software so you'll know where to go next.

® CP/M is the registered trade mark of Digital Research Inc.



Dealer enquiries invited, please contact:–

NewBrain, Grundy Business Systems Ltd., Grundy House, Somerset Road, Teddington TW11 8TD.

Each NewBrain order will include a FREE comprehensive user manual, a catalogue of expansion modules and peripherals, and a detailed list of available software.

Please send me the following:–

Quantity	Item	Price per item (inc. VAT & p&p)	Total
	NewBrain A	£233.00	
	NewBrain AD with onboard single line display	£267.50	
	Printer	£466.00	
	Monitor 12"	£142.50	
		Total £	

I enclose a cheque/Postal Order for £________ payable to Grundy Business Systems Reader Account. NewBrain, Grundy Business Systems Ltd., Grundy House, Somerset Road, Teddington TW11 8TD.

Please debit my Access Card No:________ my Barclaycard No:________

Signature________

Name________

Address________

(Dept ZX)

Postcode________

Registered Number 1522978
VAT Number 358661618
Please allow up to 28 days for delivery.

NEWBRAIN

are not lost when the ZX81 is turned off.

Syntax check

The syntax of a line of program is checked on entry. A syntax error cursor marks the first place the syntax breaks down if there is an error. The syntax error cursor disappears when errors have been corrected. Only lines free from syntax errors will be entered into the program.

Graphics

Apart from the 20 graphics characters, space and its inverse, the display may also be divided into 64 x 44 pixels, each of which may be 'blacked' in or 'whited' out under program control.

Editing

A line editor allows you to edit any line of program or input, including program line numbers. Lines may be deleted, increased or decreased in size.

Arithmetic

Arithmetic operators +, −, x, ÷, exponentiate. Relational operators =, < >, >, <,< =,>=, may compare string and arithmetic variables to yeild 0 (False) or 1(True). Logical operators AND, OR, NOT yield boolean results.

Floating-point numbers

Numbers are stored in 5 bytes in floating-point binary form giving a range of $\pm 3 \times 10^{-39}$ to $\pm 7 \times 10^{38}$ accurate to 9½ decimal digits.

Scientific functions

Natural logs/antilogs; SIN, COS, TAN and their inverses;SQR; e^x.

Variables

Numerical:	any letter followed by alphanumerics
String:	A$ to Z$
FOR-NEXT loops:	A–Z (loops may be nested to any depth.
Numerical arrays:	A–Z
String arrays:	A$ to Z$

Arrays

Arrays may be multi-dimensional with subscripts starting at 1.

Expression evaluator

The full expression evaluator is called whenever an expression, constant or variable is encountered during program execution. This powerful feature allows use of expressions in place of constants and is especially useful in GOTO, GOSUB etc.

Command mode

The ZX81 will execute statements immediately, enabling it to perform like a calculator.

Cassette interface

Works using domestic cassette recorders. The transfer rate is 250 baud and uses a unique recording format not compatible with other systems. The ZX81 will save the data as well as the program to avoid the need to re-enter the data when the program is next loaded.,

ZX81 will search through a tape for the required program). The cassette leads supplied have 3.5 mm jack plugs.

Expansion port

At the rear, this has the full data, address and control buses from the Z80A CPU as well as OV, +5V, +9V,Ø and the memory select lines. These signals enable you to interface the ZX81 to the Sinclair 16K RAM pack and ZX printer.

Power supply

The ZX81 requires approximately 420mA at 7–11V DC. It has its own internal 5V regulator. The ready assembled ZX81 comes complete with a power supply. The ZX81 kit does not include a power supply.

TV standard

The ZX81 is designed to work with UHF TVs (channel 36) 625 lines.

ZX SPECTRUM

Dimensions

Width 233 mm
Depth 144 mm
Height 30 mm

CPU/Memory

Z80A microprocessor running at 3.5 MHz. 16K-byte ROM containing BASIC interpreter and operating system.
16K-byte RAM (plus optional 32K-byte RAM on internal expansion board) or 48K-byte RAM.

Keyboard

40-key keyboard with upper and lower case with capitals lock feature. All BASIC words obtained by single keys, plus 16 graphics characters, 22 colour control codes and 21 user-definable graphics characters. All keys have auto repeat.

Display

Memory-mapped display of 256 pixels x 192 pixels; plus one attributes byte per character square, defining one of eight foreground colours, one of eight background colours, normal or extra brightness and flashing or steady. Screen border colour also settable to one of eight colours. Will drive a PAL UHF colour TV set, or black and white set (which will give a scale of grey), on channel 36.

Sound

Internal loudspeaker can be operated over more than 10 octaves (actually 130 semitones) via basic BEEP command. Jack sockets at the rear of computer allow connections to external amplifier/ speaker.

Graphics

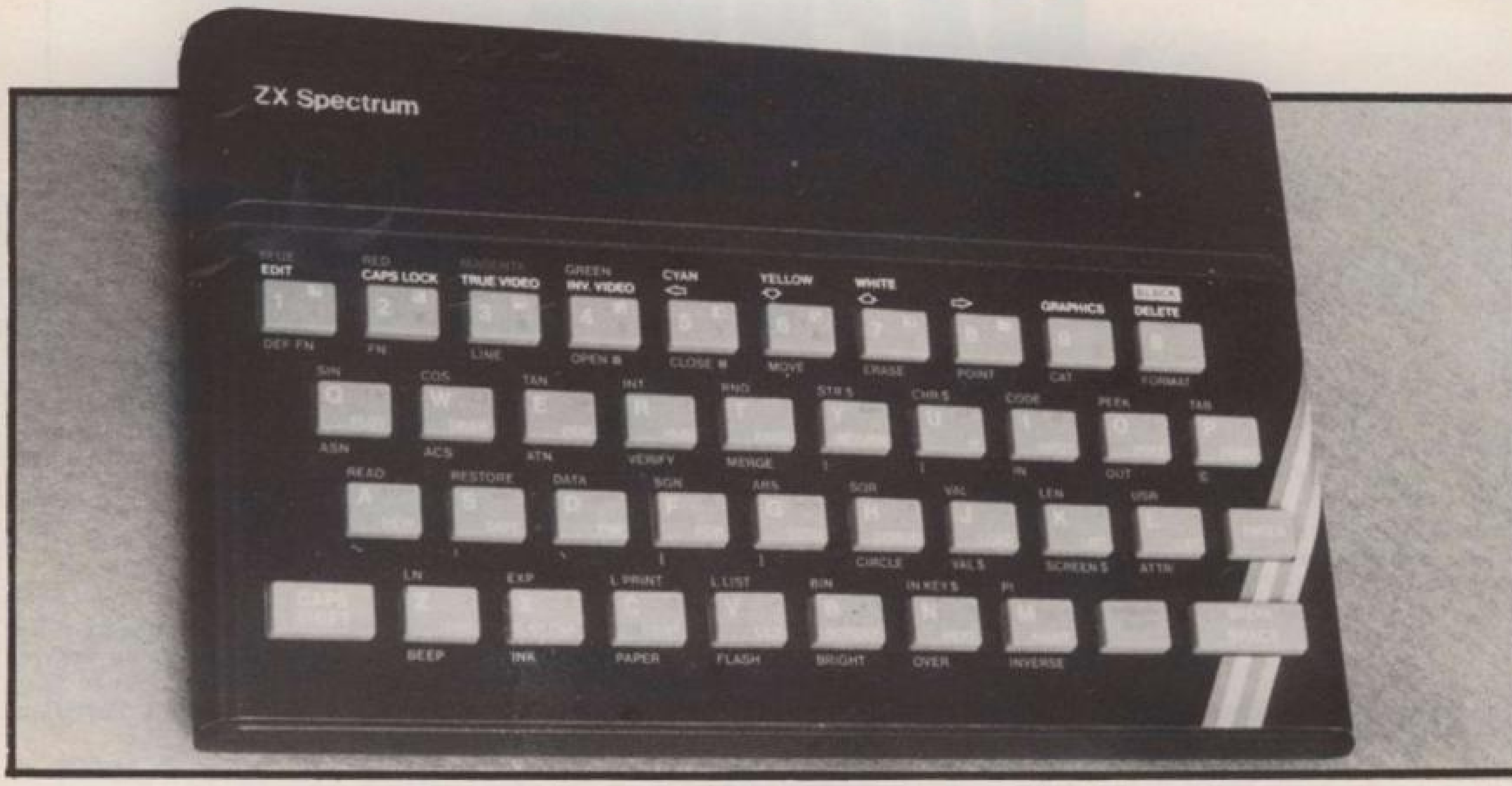
Point, line, circle and arc drawing commands in high-resolution graphics.
16 pre-defined graphics characters plus 21 user-definable graphics characters. Also functions to yield character at a given position, attribute at a given position (colours, brightness and flash) and whether a given pixel is set. Text may be written on the screen on 24 lines of 32 characters. Text and graphics may be freely mixed.

Colours

Foreground and background colours, brightness and flashing are set by BASIC INK, PAPER, BRIGHT and FLASH commands. OVER may also be set, which performs an exclusive — or operation to overwrite any printing or plotting that is already on the screen. INVERSE will give inverse video printing. These six commands may be set globally to cover all further PRINT, PLOT, DRAW or CIRCLE commands, or locally within these commands to cover only the results of that command. They may also be set locally to cover text printed by an INPUT statement. Colour-control codes, which may be accessed from the keyboard, may be inserted into text or program listing, and when displayed will override the globally set colours until another control code is encountered. Brightness and flashing codes may be inserted into program or text, similarly. Colour-control codes in a program listing have no effect on its execution. Border colour is set by a BORDER command. The eight colours available are black, blue, red, magneta, green, cyan, yellow and white. All eight colours may be present on the screen at once, with some areas flashing and others steady, and any area may be highlighted extra bright.

Screen

The screen is divided into two sections. The top section — normally the first 22 lines — displays the program listing or the results of program or command execution. The bottom section — normally the last 2 lines — shows the command or program line currently being entered, or the program line currently being edited. It also shows the report messages. Full editing facilities of cursor left, cursor right, insert and delete (with auto-repeat facility) are available over this line. The bottom section will expand to accept a current line of up to 22 lines.



Mathematical Operations And Functions

Arithmetic operations of +, −, ×, ÷, and raise to a power. Mathematical functions of sine, cosine, tangent and their inverses; natural logs and exponentials; sign function, absolute value function, and integer function; square root function, random number generation, and pi.
Numbers are stored as five bytes of floating point binary — giving a range of $+3 \times 10^{-39}$ to $+7 \times 10^{38}$ accurate to 9½ decimal digits.
Binary numbers may be entered directly with the BIN function. =, >, <, >=, <= and <> may be used to compare string or arithmetic values or variables to yield 0 (false) or 1 (true). Logical operators AND, OR and NOT yield boolean results but will accept 0 (false) and any number (true).
User-definable functions are defined using DEF FN, and called using FN. They may take up to 26 numeric and 26 string arguments, and may yield string or numeric results.
There is a full DATA mechanism, using the commands READ, DATA and RESTORE.
A real-time clock is obtainable.

String Operations And Functions

Strings can be concatenated with +. String variables or values may be compared with =, >, <, >=, <=, <> to give boolean results. String functions are VAL, VAL$, STR$ and LEN. CHR$ and CODE convert numbers to characters and vice versa, using the ASCII code. A string slicing mechanism exists, using the form a$ (x TO y).

Variable Names

Numeric — any string starting with a letter (upper and lower case are not distinguished between, and spaces are ignored).
String — A$ to Z$.
FOR-NEXT loops — A-Z.
Numeric arrays — A-Z.
String arrays — A$ to Z$.
Simple variables and arrays with the same name are allowed and distinguished between.

Arrays

Arrays may be multi-dimensional, with subscripts starting at 1. String arrays, technically character arrays, may have their last subscript omitted, yielding a string.

Expression Evaluator

A full expression evaluator is called during program execution whenever an expression, constant or variable is encountered. This allows the use of expressions as arguments to GOTO, GOSUB, etc.
It also operates on commands allowing the ZX Spectrum to operate as a calculator.

Cassette Interface

A tone leader is recorded before the information to overcome the automatic recording level fluctuations of some tape recorders, and a Schmitt trigger is used to remove noise on playback.
All saved information is started with a header containing information as to its type, title, length and address information. Program, screens, blocks of memory, string and character arrays may all be saved separately.
Programs, blocks of memory and arrays may be verified after saving.
Programs and arrays may be merged from tape to combine them with the existing contents of memory. Where two line numbers or variables names coincide, the old one is overwritten.
Programs may be saved with a line number, where execution will start immediately on loading.
The cassette interface runs at 1500 baud, through two 3.5 mm jack plugs.

Expansion Port

This has the full data, address and control busses from the Z80A, and is used to interface to the ZX Printer, the RS232 and NET interfaces and the ZX Microdrives. IN and OUT commands give the I/O port equivalents of PEEK and POKE.

ZX81 Compatibility

ZX81 BASIC is essentially a subset of ZX Spectrum BASIC. The differences are as follows.
FAST and SLOW: the ZX Spectrum operates at the speed of the ZX81 in FAST mode with the steady display of SLOW mode, and does not include these commands.
SCROLL: the ZX Spectrum scrolls automatically, asking the operator "scroll?" every time a screen is filled.
UNPLOT: the ZX Spectrum can unplot a pixel using PLOT OVER, and thus achieves unplot.
Character set: the ZX Spectrum uses the ASCII character set, as opposed to the ZX81 non-standard set.

MACHINE SPECIFICATIONS

ZX80

Dimensions
Width 174mm (6.85 in)
Depth 218mm (8.58 in)
Height 38 mm (1. 5 in)
Weight 300g (10.5oz)

Microprocessor/Memory
Z80A 3.25 MHz clock
ROM: 4K bytes containing BASIC
RAM: 1K bytes internal, externally expandable to 16K bytes.

Display
Requires an ordinary domestic black and white colour TV. The lead supplied connects between the ZX80 and your TV's aerial socket. The display organisation is 24 lines of 32 characters per line showing black characters on a white screen. The ZX80 does not connect to a printer.

Programming
Programs can be entered on the keyboard or loaded from cassette. The ZX80 has automatic "wrap round" so lines of program can be any length but not multi-statement lines.

Syntax check
The syntax of the entered line is checked character by character. A syntax error cursor marks the first place the syntax breaks down if there is an error. Once any errors have been edited out the syntax error cursor disappears. Only syntax error-free lines of code are accepted by the ZX80.

Graphics
Total of 22 graphics symbols giving 48 x 64 pixels resolution consisting of 10 symbols plus space and inverses. Includes symbols for drawing bar charts. Under control of your BASIC program any character can be printed in reverse field.

Editing
The line edit allows you to edit any line of program or input including statement numbers. The edit and cursor control keys are EDIT, RUBOUT, HOME.

Arithmetic
Arithmetic operators +,–,x, ÷ exponentiate. Relational operators <, >, = , yielding 0 or – 1. Logical operators AND OR NOT yielding boolean result. Relational operators also apply to strings. ZX80 BASIC uses 16 bit two's complement arithmetic (± 32767).

Variables
Numeric variable names may be any length, must begin with a letter and consist of alphanumerics. Every character in the name is compared thus an infinity of unique names is available.

String variables may be assigned to or from, shortened but not concatenated. String variable names are A$ – Z$. Strings do not require a dimension statement and can be any length.

Arrays have a maximum dimension of 255 (256 elements) each. Array names consist of a single letter A–Z.
Control variable names in FOR. . . NEXT loops consist of a single letter A–Z.

Expression evaluator
The full expression evaluator is called whenever a constant or variable is encountered during program execution. This allows you to use expressions in place of constants especially useful in GOTOs, GOSUBs, FOR. . . NEXT etc.

Immediate mode
The ZX80 will function in the "calculator mode" by immediately executing a statement if it is not preceded with a line number.

Cassette interface
Works with most domestic cassette recorders. The transfer rate is 250 baud using a unique tape-recording format. Other systems are not compatible with the ZX80's. The ZX80 also SAVEs the variables as well as the program on cassette. Therefore you can save the data for updating next time the program is executed. The ZX80 does not support separate data files. The lead supplied with the ZX80 is fitted with 3.5mm jack plugs.

Expansion bus
At the rear has 8 data, 16 address, 13 control lines from the processor and Ov, 5v, 9-11v, $\bar{\emptyset}$ and internal memory control line. These signals enable you to interface the ZX80 to your own electronics, PIO, CTC, SIO if you want I/O ports etc.

Power supply
The ZX80 requires approximately 400mA from 7–11v DC. It has its own internal 5v regulator.

TV standard
The ZX80 is designed to work with UHF TVs (channel 36)and is the version required for use in the United Kingdom. The ZX80 USA is designed to work with a VHF TV(American channel 2. European channel 3) and is the version required for the American TV system, also for countries without UHF.

ZX81

Dimensions
Width 167mm (6.32 in)
Depth 175mm (6.80 in)
Height 40 mm (1.57 in)
Weight 350 gms (12.15 oz)

Microprocessor/Memory
Z80A 3.25 MHz clock
ROM: Containing 8K BASIC interpreter
RAM: 1K bytes internal, externally expandable to 16K bytes.

Keyboard
40 key touch-sensitive membrane. Using function mode and single press key-word system, this gives the equivalent of 91 keys and also graphics mode allows an additional 20 graphical and 54 inverse video characters to be entered directly.

Display
Requires an ordinary domestic black and white or colour TV. The aerial lead supplied connects the ZX81 to the TV aerial socket. The display is organised as 24 lines of 32 characters with black characters on a white background.

Two mode speeds
The ZX81 can operate in two software-selectable modes - FAST and NORMAL. FAST is ideal for really high-speed computing. In NORMAL mode however the ZX81 allows continuously moving, flicker-free animated displays.

Printer
The 8K ROM will permit instructions (LPRINT, LLIST and COPY) to drive the Sinclair ZX Printer.

Programming
Programs can be entered via the keyboard or loaded from cassette. Programs and data can be saved onto cassette so that they

New From Fuller
FD System for the
ZX SPECTRUM



£39.95
+ £2.50 p & p.

Professional Keyboard & Case —

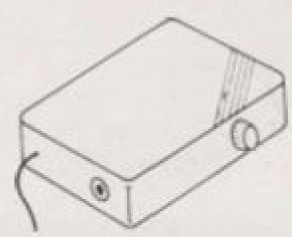
This unit has the same high standard as our ZX81 unit.
Tough A.B.S. Plastic case encloses our Keyboard, the Spectrum Printed Circuit Board and the Power Supply.
Our own Power supply is available:- 9 volts DC at 2 amps.
Mains either 110v or 240v AC at £5.95 + 80p. p & p.

The Keyboard has 42 keys with all the spectrum functions printed onto them, the full travel key switches have gold plated contacts and a guaranteed life of 10^6 operations.
INSTALLATION - Simply unscrew the ZX printed circuit board from its case and screw it into the FD case, plug in the keyboard and that's it. No technical know how or soldering required, the built unit is tested and comes with a money back guarantee.
Spectrum Keyboard and Case Kit £33.95

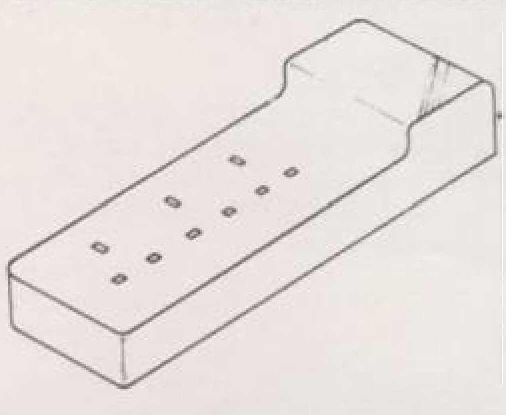
Our Mother Board for the spectrum has 2 slots at £15.95 or 3 slots at £19.95, this unit also fixes inside the case. p & p 80p.

SPECTRUM SOUND AMPLIFIER £5.95 + 80p p & p.

Complete with leads, volume control and loud speaker in tough ABS Plastic case measuring 5" x 3" x 1" just plugs into your spectrum MIC input.



SPECTRUM PLUG PLANNER — £18.95 + £1.00 p & p



Complete with 3 metres of cable, three 13 amp sockets for TV, Tape etc AND 9 volt at 2 amp power supply with power jack to fit Spectrum or ZX81

The ever popular FD42 Keyboard and case for ZX81 **£39.93** including VAT & Post
FD42 as a kit **£33.95** including VAT & Post
FD42 Built only **£24.95** including VAT & Post
FD42 Keyboard Kit **£18.95** including VAT & Post

GUARANTEED 14 DAYS DELIVERY FROM RECEIPT OF ORDER, OR CALL TO THE ZX CENTRE.

Mail to **FULLER MICRO SYSTEMS,**
The ZX Centre, Sweeting Street, Liverpool 2. England, U.K.
Please Supply:-
Name ....
Address ....

AD Code	

SAE for more details — Enquiries: Tel. 051-236 6109

FULLER FD SYSTEM



SPACE INTRUDERS WITH MUTANTS

- Four different types of Alien.
- Random saucer gives mystery points.
- Mutant Alien from second Attack-Wave.
- Bonus base at 50,000 points.
- Aliens explode when hit.
- Left, right and fire.
- Full sound effects and colour.

METEOR STORM WITH SPEECH

- Three different types of meteor.
- Two different saucers.
- Full screen wrap-around.
- Bonus ship every 10,000 points.
- Superb explosions.
- Left, right, thrust, fire and hyperspace.
- Sound effects and speech.

BOTH OF THESE FULL FEATURE GAMES INCLUDE:

- Personalised high score table.
- Hold feature, suspends game at any time.
- Progressive difficulty.
- Full "Attract Mode" with instructions and scores.

Supplied on Cassette. Both run in 16K
SEE US AT THE MICROFAIR Microscene Brum '82 show and the P.C.W. Show.
Send order to: QUICKSILVA, Dept SPB, 92 Northam Road, Southampton SO2 0PB

ORDER COUPON

Please send me:
SPACE INTRUDERS at £5.95 each
METEOR STORM(S) at £5.95 each
I enclose a cheque or P/O for

£______
(cheques payable to QUICKSILVA)

Name______

Address______

______

______